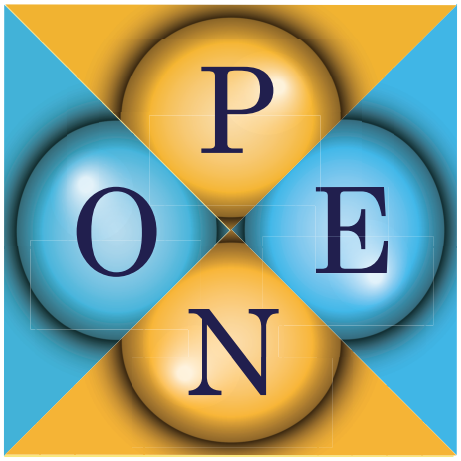
# User's manual of OpenMX Ver. 3.9




**Contributors**

T. Ozaki (Univ. of Tokyo)
H. Kino (NIMS)
J. Yu (SNU)
M.J. Han (KAIST)
M. Ohfuchi (Fujitsu Labs.)
F. Ishii (Kanazawa Univ.)
K. Sawada (Kanazawa Univ.)
Y. Kubota (Kanazawa Univ.)
Y.P. Mizuta (Osaka Univ.)
H. Kotaka (Kyoto Univ.)
N. Yamaguchi (Kanazawa Univ.)
H. Sawahata (Kanazawa Univ.)
T.B. Prayitno (Univ. Negeri Jakarta)
T. Ohwaki (NISSAN ARC)
T.V.T Duy (AISIN SEIKI)
M. Miyata (JAIST)
G. Jiang (Wuhan Univ. of Sci.&Tech.)
P.-H. Chang (George Mason Univ.)
A. Terasawa (Tokyo Tech)
Y. Gohda (Tokyo Tech)
H. Weng (CAS)
Y. Shiihara (Toyota Tech. Inst.)
M. Toyoda (Tokyo Tech.)
Y. Okuno (FUJIFILM)
R. Perez (UAM)
P.P. Bell (UAM)
M. Ellner (UAM)
Yang Xiao (NUAA)
A.M. Ito (NIFS)
M. Otani (AIST)
M. Kawamura (Univ. of Tokyo)
K. Yoshimi (Univ. of Tokyo)
C.-C. Lee (Tamkang Univ.)
Y.-T. Lee (Academia Sinica)
M. Fukuda (Univ. of Tokyo)
S. Ryee (KAIST)
K. Terakura (AIST)

December 9, 2022

# Contents

# 1 About OpenMX

OpenMX (Open source package for Material eXplorer) is a software package for nano-scale material simulations based on density functional theories (DFT) [1], norm-conserving pseudopotentials [32, 33, 34, 35, 36], and pseudo-atomic localized basis functions [41]. The methods and algorithms used in OpenMX and their implementation are carefully designed for the realization of large-scale *ab initio* electronic structure calculations on parallel computers based on the MPI or MPI/OpenMP hybrid parallelism. The efficient implementation of DFT enables us to investigate electronic, magnetic, and geometrical structures of a wide variety of materials such as bulk materials, surfaces, interfaces, liquids, and low-dimensional materials. Systems consisting of 1000 atoms can be treated using the conventional diagonalization method if several hundreds cores on a parallel computer are used. Even *ab initio* electronic structure calculations for systems consisting of more than 10000 atoms are possible with the O($N$) methods implemented in OpenMX if several thousands CPU cores on a parallel computer are available. Since optimized pseudopotentials and basis functions, which are well tested, are provided for many elements, users may be able to quickly start own calculations without preparing those data by themselves. Considerable functionalities have been implemented for calculations of physical properties such as magnetic, dielectric, and electric transport properties. Thus, it is expected that OpenMX can be a useful and powerful theoretical tool for nano-scale material sciences, leading to better and deeper understanding of complicated and useful materials based on quantum mechanics. The development of OpenMX has been initiated by the Ozaki group in 2000, and from then onward many developers listed in the top page of the manual have contributed for further development of the open source package. The distribution of the program package and the source codes follow the practice of the GNU General Public License version 3 (GPLv3) [102], and they are downloadable from http://www.openmx-square.org/

Features and capabilities of OpenMX Ver. 3.9 are listed below:

- total energy and forces by cluster, band, O($N$), and low-order scaling methods
- local density approximation (LDA, LSDA) [2, 3, 4] and generalized gradient approximation (GGA) [5] to the exchange-correlation potential
- DFT+U methods [20]
- norm-conserving pseudopotentials [2, 33, 34, 36]
- variationally optimized pseudo-atomic basis functions [41]
- fully and scalar relativistic treatment within pseudopotential scheme [12, 32, 16]
- non-collinear DFT [8, 9, 10, 11]
- constraint DFT for non-collinear spin and orbital orientation [13]
- macroscopic polarization by Berry's phase [15]
- O($N$) divide-conquer (DC) method [50]
- O($N$) divide-conquer with localized natural orbitals (DC-LNO) method [51]

- O($N$) Krylov subspace method [43]
- parallel eigensolver by ELPA [39]
- simple, RMM-DIIS [58], GR-Pulay [57], Kerker [59], RMM-DIIS with Kerker's metric [58], and RMM-DIIS for Hamiltonian matrix [58] charge mixing schemes
- exchange coupling parameter [17, 18]
- effective screening medium (ESM) method [125, 128]
- scanning tunneling microscope (STM) simulation [71]
- DFT-D2 and DFT-D3 method for vdW interaction [135, 136, 137]
- unfolding method for band structures [142]
- nudged elastic band (NEB) method [72]
- calculations of absolute binding energies of core levels in bulks [88]
- optical conductivity and dielectric function [98]
- charge doping
- uniform electric field
- orbitally decomposed total energy
- fully and constrained geometry optimization
- fully and constrained variable cell optimization
- electric transport calculations by a non-equilibrium Green's function (NEGF) method [73]
- construction of maximally localized Wannier functions
- NVE ensemble molecular dynamics
- NVT ensemble molecular dynamics by a velocity scaling [30] and the Nose-Hoover methods [31]
- Mulliken, Voronoi, and ESP fitting analysis of charge and spin densities
- natural population analysis [7]
- analysis of wave functions and electron (spin) densities
- dispersion analysis by the band calculation
- density of states (DOS) and projected DOS
- flexible data format for the input
- interface with BoltzTrap [100, 101]
- interface with Wannier90 [145]

- interface with XCrySDen for visualizing data such as charge density [105]
- completely dynamic memory allocation
- parallel execution by Message Passing Interface (MPI)
- parallel execution by OpenMP
- useful user interface for developers

The collinear and non-collinear (NC) DFT methods are implemented including scalar and fully relativistic pseudopotentials, respectively. The constraint NC-DFT is also supported to control spin and orbital magnetic moments. These methods will be useful to investigate complicated NC magnetic structures and the effect of spin-orbit coupling. The diagonalization of the conventional calculations is performed by a ELPA based parallel eigensolver [39] and ScaLAPACK which scales up to several thousands cores. The feature may allow us to investigate systems consisting of 1000 atoms using the conventional diagonalization. Not only the conventional diagonalization scheme is provided for clusters, molecules, slab, and solids, but also linear scaling and a low-order scaling methods are supported as eigenvalue solver. With a proper choice for the eigenvalue solvers, systems consisting of more than 10000 atoms can be treated with careful consideration to balance between accuracy and efficiency. The variable cell optimization and band unfolding method are available. As new important features of OpenMX Ver. 3.9, it is worth mentioning that we release a novel O($N$) method based on divide-conqure approach and localized natural orbitals, and calculations of absolute binding energies of core levels in bulks, which can be directly compared to binding energies observed in X-ray photoemission spectroscopy (XPS). We are continuously working toward development. Motivated contributors who want to develop the open source codes are always welcome.

## 2 Related papers

OpenMX is distributed under the terms of the GNU General Public License Version 3 (GPLv3) [102]. However, we would like to appreciate your citation of the following papers when you publish a paper using the corresponding functionality in OpenMX, which would be an implicit cooperation for the future development of OpenMX.

- General

  T. Ozaki, Phys. Rev. B. **67**, 155108, (2003); T. Ozaki and H. Kino, Phys. Rev. B **69**, 195113 (2004); T. Ozaki and H. Kino, Phys. Rev. B **72**, 045121 (2005); K. Lejaeghere et al., Science 351, aad3000 (2016).

- Large-scale parallel calculation

  T.V.T. Duy and T. Ozaki, Comput. Phys. Commun. **185**, 777 (2014); T.V.T. Duy and T. Ozaki, Comput. Phys. Commun. **185**, 153 (2014).

- O($N$) DC-LNO method

  T. Ozaki, M. Fukuda, G. Jiang, Phys. Rev. B **98**, 245137 (2018).

- O($N$) Krylov subspace method

  T. Ozaki, Phys. Rev. B **74**, 245101 (2006).

- Numerically exact low-order scaling method

  T. Ozaki, Phys. Rev. B **82**, 075131 (2010).

- The DFT+U methods

  M.J. Han, T. Ozaki, and J. Yu, Phys. Rev. B **74**, 045110 (2006); S. Ryee and M.J. Han, J. Phys:Condens. Matter **30**, 275802 (2018). S. Ryee and M.J. Han, Scientific Reports **8**, 9559 (2018).

- Exchange coupling parameter

  M.J. Han, T. Ozaki, and J. Yu, Phys. Rev. B **70**, 184421 (2004); A Terasawa, M Matsumoto, T Ozaki, and Y Gohda, J. Phys. Soc. Jpn. **88**, 114706 (2019).

- NEGF method

  T. Ozaki, K. Nishio, and H. Kino, Phys. Rev. **81**, 035116 (2010); T. Ozaki, Phys. Rev. B **75**, 035123 (2007).

- Effective screening medium method

  T. Ohwaki, M. Otani, T. Ikeshoji, and T. Ozaki, J. Chem. Phys. 136, 134101 (2012).

- Generation of Wannier functions

  H. Weng, T. Ozaki, and K. Terakura, Phys. Rev. B **79**, 235118 (2009).

- Generation of natural atomic orbitals

  T. Ohwaki, M. Otani, and T. Ozaki, J. Chem. Phys. 140, 244105 (2014).

- Band unfolding method

  C.-C. Lee, Y. Yamada-Takamura, and T. Ozaki, J. Phys.: Condens. Matter **25**, 345501 (2013).

- XPS binding energies

  T. Ozaki and C.-C. Lee, Phys. Rev. Lett. **118**, 026401 (2017).

- BoltzTraP calculations

  M. Miyata, T. Ozaki, T. Takeuchi, S. Nishino, M. Inukai, and M. Koyano, Journal of Electronic Materials **47**, 3254 (2017).

- FermiSurfer

  M. Kawamura, Comp. Phys. Comm. **239**, 197 (2019).

- Analysis of spin texture in the k-space

  H. Kotaka, F. Ishii, and M. Saito, Jpn. J. Appl. Phys. **52**, 035204 (2013).; N. Yamaguchi and F. Ishii, Appl. Phys. Express **10**, 123003 (2017).

- Spin spiral calculations

  T.B. Prayitno and F. Ishii, J. Phys. Soc. Jpn. **87**, 114709 (2018); T.B. Prayitno and F. Ishii, J. Phys. Soc. Jpn. **88**, 054701 (2019).

- $Z_2$, Chern number, and Berry curvature

  H. Sawahata, N. Yamaguchi, H. Kotaka, and F. Ishii, Jpn. J. Appl. Phys. **57**, 030309 (2018).

- OpenMX Viewer

  Y.-T Lee and T. Ozaki, Journal of Molecular Graphics and Modelling **89**, 192 (2019).

# 3 Installation

## 3.1 Including libraries

OpenMX can be installed under linux environment where three library packages are available as listed below:

- ScaLAPACK (and BLACS) (http://www.netlib.org/)
- FFTW (http://www.fftw.org/)
- MPI library such as MPICH2 and OpenMPI

If these library packages are not installed on your machine, you are required to install them before the installation of OpenMX. Note that a MPI library such as MPICH2 and OpenMPI has to be available for the installation of OpenMX Ver. 3.9. Without a MPI library, OpenMX Ver. 3.9 cannot be installed. Also, OpenMX Ver. 3.9 requires ScaLAPACK and BLACS, and the compilation of OpenMX Ver. 3.9 with LAPACK and BLAS is not supported. As an alternative, the Intel Math Kernel Library (MKL) can also be utilized instead of the naive ScaLAPACK and BLACS. If these libraries packages are available on your machine, you can proceed the following procedure for the installation. Then, after downloading 'openmx3.9.tar.gz', decompress it as follows:

```
% tar zxvf openmx3.9.tar.gz
```

When it is completed, you can find three directories 'source', 'work', 'DFT_DATA19' under the directory 'openmx3.9'. The directories 'source', 'work', and 'DFT_DATA19' contain source files, input files, and data files for optimized pseudo-atomic basis functions and pseudopotentials of Ver. 2019, respectively.

## 3.2 Serial version

The installation of the serial version is not supported for OpenMX Ver. 3.9.

## 3.3 MPI version

To proceed the installation of the MPI version, move to the directory 'source', and modify 'makefile' in 'source' to specify the compiler and libraries by **CC**, **FC**, and **LIB**. The default specification of **CC**, **FC**, and **LIB** in 'makefile' is as follows:

```
MKLROOT = /opt/intel/mkl
CC = mpicc -O3 -xHOST -ip -no-prec-div -qopenmp -I/opt/intel/mkl/include/fftw
FC = mpif90 -O3 -xHOST -ip -no-prec-div -qopenmp
LIB= -L${MKLROOT}/lib/intel64 -lmkl_scalapack_lp64 -lmkl_intel_lp64 \
     -lmkl_intel_thread -lmkl_core -lmkl_blacs_open mpi_lp64 \
     -lmpi_usempif08 -lmpi_usempi_ignore_tkr \
     -lmpi_mpifh -liomp5 -lpthread -lm -ldl
```

**CC** and **FC** are the specification for C and FORTRAN compilers, respectively, and **LIB** is the specification for libraries which are linked. Although the specification of **FC** is not required up to and including Ver. 3.6, **FC** must be specified in Ver. 3.9 due to the introduction of the ELPA based parallel eigensolver [39]. The option '-Dnoomp' should be added under environment that OpenMP is not available. You need to set the **CC**, **FC** and **LIB** appropriately on your computer environment so that the compilation and linking can be properly performed and the executable file can be well optimized, while the specification largely depends on your computer environment. After specifying **CC**, **FC** and **LIB** appropriately, then install as follows:

```
% make install
```

When the compilation is completed normally, then you can find the executable file 'openmx' in the directory 'work'. To make the execution of OpenMX efficient, you can change a compiler and compile options appropriate for your computer environment, which can generate an optimized executable file. Several examples for **CC**, **FC** and **LIB** can be found in 'makefile' in the directory 'source' for your convenience.

### 3.4 MPI/OpenMP version

To generate the MPI/OpenMP hybrid version, all you have to do is to include a compiler option for OpenMP parallelization for **CC** and **FC** in 'makefile' in the directory 'source'. To proceed the installation of the MPI/OpenMP version, move to the directory 'source', and specify **CC**, **FC** and **LIB** in 'makefile', for example, as follows:

For icc

```
CC = mpicc -O3 -xHOST -ip -no-prec-div -qopenmp -I/opt/intel/mkl/include/fftw
FC = mpif90 -O3 -xHOST -ip -no-prec-div -qopenmp
```

Note that the compiler option for OpenMP depends on compiler, while the option corresponds to '-qopenmp' for the Intel compiler, After specifying **CC**, **FC**, and **LIB** appropriately, then install as follows:

```
% make install
```

When the compilation is completed normally, then you can find the executable file 'openmx' in the directory 'work'. To make the execution of OpenMX efficient, you can change a compiler and compile options appropriate for your computer environment, which can generate an optimized executable file.

### 3.5 FFTW3

OpenMX Ver. 3.9 supports only FFTW3, while older versions up to Ver. 3.6 also support FFTW2 as well as FFTW3. Then, you may link FFTW3 in your makefile as follows:

```
LIB = -lfftw3
```

We wonder that many users will use the built-in libarary of FFTW3 in the Intel MKL.

## 3.6 Other options

### 3.6.1 -Dnosse

Since the routine (Krylov.c) for the O($N$) Krylov subspace method has been optimized using Streaming SIMD Extensions (SSE), the code will be compiled including SSE on default compilation. If your processors do not support SSE, then include '-Dnosse' as compilation option for **CC**.

### 3.6.2 -Dkcomp

For SPARC processors developed by FUJITSU Ltd., include -Dkcomp as compilation option for **CC** and **FC**.

## 3.7 Platforms

So far, we have confirmed that OpenMX Ver. 3.9 runs normally on the following machines:

- Intel Xeon based clusters
- AMD EPYC based clusters
- CRAY-XC40
- Fujitsu FX100

## 3.8 Tips for installation

Most problems in the installation of OpenMX are caused by the linking of ScaLAPACK and BLACS or its alternative. We would recommend users to link MKL in most cases. Examples on how to link them can be found in 'makefile' in the directory 'source'.

Also, we provide a couple of tips for the installation on popular platforms below. OpenMX requires C and FORTRAN compilers, ScaLAPACK and BLACS libraries, and FFTW3 library. In addition, as the C compiler is used for linking, the corresponding FORTRAN library of the compiler should be explicitly specified. Here we provide some sample settings for installation on platforms with several popular compilers and ScaLAPACK and BLACS libraries under an assumption that the FFT library is installed in /usr/local/fftw3/.

- Intel C and FORTRAN compilers (icc, ifort) and the MKL library for ScaLAPACK and BLACS

  MKLROOT = /opt/intel/mkl
  CC = mpicc -O3 -xHOST -ip -no-prec-div -qopenmp -I/usr/local/fftw3/include
  FC = mpif90 -O3 -xHOST -ip -no-prec-div -qopenmp
  LIB= -L/usr/local/fftw3/lib -lfftw3 \
  -L$MKLROOT/lib/intel64 -lmkl_scalapack_lp64 -lmkl_intel_lp64 \
  -lmkl_intel_thread -lmkl_core -lmkl_blacs_open mpi_lp64 \
  -lmpi_usempif08 -lmpi_usempi_ignore_tkr \
  -lmpi_mpifh -liomp5 -lpthread -lm -ldl

- GNU C and FORTRAN compilers (gcc, g++, gfortran) and the MKL library for ScaLAPACK and BLACS

  MKLROOT=/opt/intel/mkl
  CC = mpicc -O3 -ffast-math -fopenmp -I/usr/local/fftw3/include -I/$MKLROOT/include
  FC = mpif90 -O3 -ffast-math -fopenmp -I/$MKLROOT/include
  LIB= -L/usr/local/fftw3/lib -lfftw3 \
  -L$MKLROOT/lib/intel64 -lmkl_scalapack_lp64 -lmkl_intel_lp64 \
  -lmkl_intel_thread -lmkl_core -lmkl_blacs_open mpi_lp64 \
  -lmpi_usempif08 -lmpi_usempi_ignore_tkr \
  -lmpi_mpifh -liomp5 -lpthread -lm -ldl

- GNU C and FORTRAN compilers (gcc, g++, gfortran) and ScaLAPACK and BLACS

  CC = mpicc -O3 -ffast-math -fopenmp -Dkcomp -I/usr/local/include -I/home/ytl/openmpi-3.0.1/ompi/include
  FC = mpif90 -O3 -ffast-math -fopenmp -Dkcomp -I/home/ytl/openmpi-3.0.1/ompi/include
  LIB= -L/usr/local/lib -lfftw3 -L/home/ytl/openmpi-3.0.1/ompi -lmpi -lmpi_mpifh \
  -L/home/ytl/Packages/lapack-3.7.0 -llapack -lrefblas -lgfortran

- FUJITSU compilers on FX100 machines

  CC = mpifccpx -Kfast -Dnosse -Dkcomp
  FC = mpifrtpx -Kfast -Dkcomp
  LIB = -lfftw3 -SCALAPACK -SSL2BLAMP

Other combinations of the compiler and ScaLAPACK and BLACS libraries can be done in the same fashion. The following commands can be used to show information about the compiler (Intel, PGI, GNU, etc.) used by MPI.

```
%mpicc -compile-info (with MPICH)
%mpicc -help  (with OpenMPI)
```

In some cases, the location of the FORTRAN library is unknown to the C compiler, resulting in the following link errors:

```
/usr/bin/ld: cannot find -lifcore
```

with the Intel compiler,

```
/usr/bin/ld: cannot find -lpgf90
```

with the PGI compiler, or

```
-lpgf90_rpm1, -lpgf902, -lpgf90rtl, -lpgftnrtl
```

as the "-pgf90libs" flag is just a shortcut for them,

```
/usr/bin/ld: cannot find -lgfortran
```

with the GNU compiler.

To solve this link-time problem, the location of the FORTRAN library must be explicitly specified as follows. First, the location of the FORTRAN compiler can be identified with the following commands.

```
%which ifort (with the Intel compiler)
/opt/intel/fce/10.0.026/bin/ifort

%which pgf90 (with the PGI compiler)
/opt/pgi/linux86-64/7.0/bin/pgf90

%which gfortran (with the GNU compiler)
/usr/bin/gfortran
```

Then, the location of the FORTRAN library usually resides in /lib of the parent folder of /bin, and must be specified in LIB such as

```
LIB= ... -L/opt/intel/fce/10.0.026/lib -lifcore (with the Intel compiler)
LIB= ... -L/opt/pgi/linux86-64/7.0/lib -pgf90libs (with the PGI compiler)
LIB= ... -L/usr/lib -lgfortran (with the GNU compiler)
```

As for the Intel Math Kernel Library, you may use the folowing website: Intel Math Kernel Library Link Line Advisor

```
https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor
```

which gives us a suggestion for linking of libraries and compiler options.

### 3.9 Options for *make*

All the object and executable files can be cleaned as follows:

```
% make clean
```

All the executable files including post-processing codes can be compiled as follows:

```
% make all
```

# 4 Test calculation

If the installation is completed normally, please move to the directory 'work' and perform the program 'openmx' using an input file 'Methane.dat' which can be found in the directory 'work' as follows:

```
% mpirun -np 1 openmx Methane.dat > met.std &
```

Or if you use the MPI/OpenMP version:

```
% mpirun -np 1 openmx Methane.dat -nt 1 > met.std &
```

The test input file 'Methane.dat' is for performing the SCF calculation of a methane molecule with a fixed structure (No MD). The calculation is performed in only about 5 seconds by using a single core on a 2.6 GHz Xeon machine, although it is dependent on a computer. When the calculation is completed normally, 11 files and one directory

```
met.std              standard output of the SCF calculation
met.out              input file and standard output
met.xyz              final geometrical structure
met.ene              values computed at every MD step
met.md               geometrical structures at every MD step
met.md2              geometrical structure of the final MD step
met.cif              cif file of the initial structure for Material Studio
met.tden.cube        total electron density in the Gaussian cube format
met.v0.cube          Kohn-Sham potential in the Gaussian cube format
met.vhart.cube       Hartree potential in the Gaussian cube format
met.dden.cube        difference electron density measured from atomic density
met_rst/             directory storing restart files
```

are output to the directory 'work'. The output data to a standard output is stored to the file 'met.std' which is helpful to know the computational flow of the SCF procedure. The file 'met.out' includes computed results such as the total energy, forces, the Kohn-Sham eigenvalues, Mulliken charges, the convergence history for the SCF calculation, and analyzed computational time. A part of the file 'met.out' is shown below. It is found that the eigenvalues energy converges by 14 iterations within 1.0e-10 Hartree.

```
*******************************************************
*******************************************************
                  SCF history at MD= 1
*******************************************************
*******************************************************

   SCF=   1  NormRD=  1.000000000000  Uele= -3.523169099731
   SCF=   2  NormRD=  0.181517404404  Uele= -3.686855123738
   SCF=   3  NormRD=  0.449067381009  Uele= -4.193062144919
   SCF=   4  NormRD=  0.541215648203  Uele= -4.381387140154
   SCF=   5  NormRD=  0.509921689399  Uele= -4.352426233337
```

```
 SCF=   6  NormRD=  0.004026023243  Uele= -3.886371199720
 SCF=   7  NormRD=  0.000838640096  Uele= -3.889312346884
 SCF=   8  NormRD=  0.000420666755  Uele= -3.889396659132
 SCF=   9  NormRD=  0.000241013350  Uele= -3.889362708861
 SCF=  10  NormRD=  0.000573725679  Uele= -3.889427222948
 SCF=  11  NormRD=  0.000000150516  Uele= -3.889316043314
 SCF=  12  NormRD=  0.000000001917  Uele= -3.889316014533
 SCF=  13  NormRD=  0.000000000005  Uele= -3.889316014156
 SCF=  14  NormRD=  0.000000000001  Uele= -3.889316014146
```

Also, the total energy, chemical potential, Kohn-Sham eigenvalues, the Mulliken charges, dipole moment, forces, fractional coordinate, and analysis of computational time are output in 'met.out' as follows:

```
*******************************************************
        Total energy (Hartree) at MD = 1
*******************************************************

  Uele.         -3.889316014146

  Ukin.          5.533759381370
  UH0.         -14.855519969177
  UH1.           0.041396138425
  Una.          -5.040606545149
  Unl.          -0.134650846490
  Uxc0.         -1.564720263874
  Uxc1.         -1.564720263874
  Ucore.         9.551521413583
  Uhub.          0.000000000000
  Ucs.           0.000000000000
  Uzs.           0.000000000000
  Uzo.           0.000000000000
  Uef.           0.000000000000
  UvdW           0.000000000000
  Uch            0.000000000000
  Utot.         -8.033540955187

  Note:

  Utot = Ukin+UH0+UH1+Una+Unl+Uxc0+Uxc1+Ucore+Uhub+Ucs+Uzs+Uzo+Uef+UvdW

  Uene:   band energy
  Ukin:   kinetic energy
  UH0:    electric part of screened Coulomb energy
```

```
  UH1:    difference electron-electron Coulomb energy
  Una:    neutral atom potential energy
  Unl:    non-local potential energy
  Uxc0:   exchange-correlation energy for alpha spin
  Uxc1:   exchange-correlation energy for beta spin
  Ucore:  core-core Coulomb energy
  Uhub:   DFT+U energy
  Ucs:    constraint energy for spin orientation
  Uzs:    Zeeman term for spin magnetic moment
  Uzo:    Zeeman term for orbital magnetic moment
  Uef:    electric energy by electric field
  UvdW:   semi-empirical vdW energy

  (see also PRB 72, 045121(2005) for the energy contributions)


  Chemical potential (Hartree)       0.000000000000

***********************************************************
***********************************************************
           Eigenvalues (Hartree) for SCF KS-eq.
***********************************************************
***********************************************************

   Chemical Potential (Hartree) =   0.00000000000000
   Number of States             =   8.00000000000000
   HOMO =  4
   Eigenvalues
                Up-spin            Down-spin
          1  -0.69897506408475  -0.69897506408475
          2  -0.41523055776668  -0.41523055776668
          3  -0.41523055768741  -0.41523055768741
          4  -0.41522182758055  -0.41522182758055
          5   0.21221759603691   0.21221759603691
          6   0.21221759685634   0.21221759685634
          7   0.21230533059490   0.21230533059490
          8   0.24741918440773   0.24741918440773

***********************************************************
***********************************************************
                   Mulliken populations
***********************************************************
***********************************************************
```

```
 Total spin moment (muB)   0.000000000

                    Up spin      Down spin     Sum           Diff
     1    C      2.509748760  2.509748760   5.019497520   0.000000000
     2    H      0.372562810  0.372562810   0.745125620   0.000000000
     3    H      0.372562810  0.372562810   0.745125620   0.000000000
     4    H      0.372562810  0.372562810   0.745125620   0.000000000
     5    H      0.372562810  0.372562810   0.745125620   0.000000000

Sum of MulP: up   =     4.00000 down          =     4.00000
             total=     8.00000 ideal(neutral)=     8.00000


 Decomposed Mulliken populations
   1    C          Up spin      Down spin     Sum           Diff
          multiple
 s           0    0.681737894  0.681737894   1.363475787   0.000000000
  sum over m      0.681737894  0.681737894   1.363475787   0.000000000
  sum over m+mul  0.681737894  0.681737894   1.363475787   0.000000000
 px           0    0.609352701  0.609352701   1.218705403   0.000000000
 py           0    0.609305463  0.609305463   1.218610926   0.000000000
 pz           0    0.609352702  0.609352702   1.218705404   0.000000000
  sum over m      1.828010866  1.828010866   3.656021733   0.000000000
  sum over m+mul  1.828010866  1.828010866   3.656021733   0.000000000

   2    H          Up spin      Down spin     Sum           Diff
          multiple
 s           0    0.372562810  0.372562810   0.745125620   0.000000000
  sum over m      0.372562810  0.372562810   0.745125620   0.000000000
  sum over m+mul  0.372562810  0.372562810   0.745125620   0.000000000

   3    H          Up spin      Down spin     Sum           Diff
          multiple
 s           0    0.372562810  0.372562810   0.745125620   0.000000000
  sum over m      0.372562810  0.372562810   0.745125620   0.000000000
  sum over m+mul  0.372562810  0.372562810   0.745125620   0.000000000

   4    H          Up spin      Down spin     Sum           Diff
          multiple
 s           0    0.372562810  0.372562810   0.745125620   0.000000000
  sum over m      0.372562810  0.372562810   0.745125620   0.000000000
  sum over m+mul  0.372562810  0.372562810   0.745125620   0.000000000

   5    H          Up spin      Down spin     Sum           Diff
```

```
            multiple
  s           0    0.372562810  0.372562810   0.745125620   0.000000000
   sum over m      0.372562810  0.372562810   0.745125620   0.000000000
   sum over m+mul  0.372562810  0.372562810   0.745125620   0.000000000

************************************************************
************************************************************
                    Dipole moment (Debye)
************************************************************
************************************************************

 Absolute D        0.00000000

                      Dx               Dy               Dz
 Total              0.00000000       0.00000000       0.00000000
 Core               0.00000000       0.00000000       0.00000000
 Electron           0.00000000       0.00000000       0.00000000
 Back ground       -0.00000000      -0.00000000      -0.00000000

************************************************************
************************************************************
       xyz-coordinates (Ang) and forces (Hartree/Bohr)
************************************************************
************************************************************

<coordinates.forces
  5
    1     C     0.00000   0.00000   0.00000   0.000000000000  0.00...
    2     H    -0.88998  -0.62931   0.00000  -0.064890985127 -0.04...
    3     H     0.00000   0.62931  -0.88998   0.000000000002  0.04...
    4     H     0.00000   0.62931   0.88998   0.000000000002  0.04...
    5     H     0.88998  -0.62931   0.00000   0.064890985122 -0.04...
coordinates.forces>

************************************************************
************************************************************
       Fractional coordinates of the final structure
************************************************************
************************************************************

     1      C     0.00000000000000   0.00000000000000   0.00000000000000
     2      H     0.91100190000000   0.93706880000000   0.00000000000000
     3      H     0.00000000000000   0.06293120000000   0.91100190000000
     4      H     0.00000000000000   0.06293120000000   0.08899810000000
```

```
     5      H     0.08899810000000   0.93706880000000   0.00000000000000

*************************************************************
*************************************************************
               Computational Time (second)
*************************************************************
*************************************************************

   Elapsed.Time.         4.600

                                Min_ID   Min_Time       Max_ID   Max_Time
   Total Computational Time =     0        4.600          0        4.600
   readfile                 =     0        2.578          0        2.578
   truncation               =     0        0.146          0        0.146
   MD_pac                   =     0        0.000          0        0.000
   OutData                  =     0        0.283          0        0.283
   DFT                      =     0        1.591          0        1.591

*** In DFT ***

   Set_OLP_Kin              =     0        0.052          0        0.052
   Set_Nonlocal             =     0        0.039          0        0.039
   Set_ProExpn_VNA          =     0        0.156          0        0.156
   Set_Hamiltonian          =     0        0.663          0        0.663
   Poisson                  =     0        0.214          0        0.214
   Diagonalization          =     0        0.005          0        0.005
   Mixing_DM                =     0        0.000          0        0.000
   Force                    =     0        0.039          0        0.039
   Total_Energy             =     0        0.256          0        0.256
   Set_Aden_Grid            =     0        0.019          0        0.019
   Set_Orbitals_Grid        =     0        0.015          0        0.015
   Set_Density_Grid         =     0        0.124          0        0.124
   RestartFileDFT           =     0        0.004          0        0.004
   Mulliken_Charge          =     0        0.000          0        0.000
   FFT(2D)_Density          =     0        0.000          0        0.000
   Others                   =     0        0.005          0        0.005
```

The files 'met.tden.cube', 'met.v0.cube', 'met.vhart.cube', and 'met.dden.cube', are the total electron density, the Kohn-Sham potential, the Hartree potential, and the difference electron density taken from the superposition of atomic densities of constituent atoms, respectively, which are output in the Gaussian cube format. Since the Gaussian cube format is one of well used grid formats, you can visualize the files using free molecular modeling software such as VESTA [103], Molekel [104], and XCrySDen [105]. The visualization will be illustrated in the later section.

# 5 Automatic running test

In addition to a running test of the Section 'Test calculation', if you want to check whether most functionalities of OpenMX have been successfully installed on your computer or not, we recommend for you to perform an automatic running test. To do this, you can run OpenMX as follows:

### For the MPI parallel running

```
% mpirun -np 8 openmx -runtest
```

### For the MPI/OpenMP parallel running

```
% mpirun -np 8 openmx -runtest -nt 2
```

In the parallel execution, you can specify other options for mpirun. Then, OpenMX will run with 14 test files, and compare calculated results with the reference results which are stored in 'work/input_example'. The comparison (absolute difference in the total energy and force) is stored in a file 'runtest.result' in the directory 'work'. The reference results were calculated using a single processor of a 2.6 GHz Xeon machine. If the difference is within last seven digits, we may consider that the installation is successful. As an example, 'runtest.result' generated by the automatic running test is shown below:

mx17 (Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz), a cluster machine in the Ozaki laboratory
icc version 16.0.2, compiler option -O3 -xHOST -ip -no-prec-div -qopenmp
20 processes (MPI) x 1 thread (OpenMP)

| | | | | |
|---|---|---|---|---|
| 1 | input_example/Benzene.dat | Elapsed time(s)= 3.09 | diff Utot= 0.000000000021 | diff Force= 0.000000000000 |
| 2 | input_example/C60.dat | Elapsed time(s)= 8.71 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |
| 3 | input_example/CO.dat | Elapsed time(s)= 6.77 | diff Utot= 0.000000000000 | diff Force= 0.000000000004 |
| 4 | input_example/Cr2.dat | Elapsed time(s)= 7.15 | diff Utot= 0.000000000001 | diff Force= 0.000000000001 |
| 5 | input_example/Crys-MnO.dat | Elapsed time(s)= 20.16 | diff Utot= 0.000000000003 | diff Force= 0.000000000047 |
| 6 | input_example/GaAs.dat | Elapsed time(s)= 27.89 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |
| 7 | input_example/Glycine.dat | Elapsed time(s)= 3.45 | diff Utot= 0.000000000001 | diff Force= 0.000000000000 |
| 8 | input_example/Graphite4.dat | Elapsed time(s)= 3.75 | diff Utot= 0.000000000004 | diff Force= 0.000000000152 |
| 9 | input_example/H2O-EF.dat | Elapsed time(s)= 2.93 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |
| 10 | input_example/H2O.dat | Elapsed time(s)= 2.80 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |
| 11 | input_example/HMn.dat | Elapsed time(s)= 8.84 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |
| 12 | input_example/Methane.dat | Elapsed time(s)= 2.38 | diff Utot= 0.000000000013 | diff Force= 0.000000000001 |
| 13 | input_example/Mol_MnO.dat | Elapsed time(s)= 6.05 | diff Utot= 0.000000000001 | diff Force= 0.000000000000 |
| 14 | input_example/Ndia2.dat | Elapsed time(s)= 4.98 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |

Total elapsed time (s) 108.96

The comparison was made using 20 MPI processes on the same Xeon cluster machine as used for the calculations to obtain the reference results. Since the floating point operation depends on not only computer environment, but also the number of processors used in parallel execution, we see in the above example that there is a small difference even using the same machine. The elapsed time of each job is also output, so it is helpful in comparing the computational speed depending on computer environment. In the directory 'work/input_example', you can find 'runtest.result' files generated on several platforms.

If you want to make reference files by yourself, please execute OpenMX as follows:

```
% ./openmx -maketest
```

Then, for input files '*.dat' in the directory 'work/input_example', OpenMX will generate the output files '*.out' in 'work/input_example'. So, you can add a new dat file which is used in the next running test. But, please make sure that the previous out files in 'work/input_example' will be overwritten by this procedure. For advanced testers for checking the reliability of code, see also the Sections 'Automatic force tester' and 'Automatic memory leak tester'.

For reference, the results of 'runtest' on a couple of machines are given below:

System B (sekirei) at ISSP, Univ. of Tokyo (Intel Xeon E5-2680v3 12core 2.5GHz)
icc version 18.0.5, compiler option: -O3 -xHOST -ip -no-prec-div -qopenmp -Dkcomp -fp-model precise
6 processes (MPI) x 4 thread (OpenMP)

| | | | | |
|---|---|---|---|---|
| 1 | input_example/Benzene.dat | Elapsed time(s)= 3.69 | diff Utot= 0.000000000034 | diff Force= 0.000000000005 |
| 2 | input_example/C60.dat | Elapsed time(s)= 11.54 | diff Utot= 0.000000000005 | diff Force= 0.000000000006 |
| 3 | input_example/CO.dat | Elapsed time(s)= 6.14 | diff Utot= 0.000000000106 | diff Force= 0.000000001979 |
| 4 | input_example/Cr2.dat | Elapsed time(s)= 5.80 | diff Utot= 0.000000000364 | diff Force= 0.000000000033 |
| 5 | input_example/Crys-MnO.dat | Elapsed time(s)= 58.73 | diff Utot= 0.000000000003 | diff Force= 0.000000000005 |
| 6 | input_example/GaAs.dat | Elapsed time(s)= 48.60 | diff Utot= 0.000000000010 | diff Force= 0.000000000002 |
| 7 | input_example/Glycine.dat | Elapsed time(s)= 3.39 | diff Utot= 0.000000000001 | diff Force= 0.000000000000 |
| 8 | input_example/Graphite4.dat | Elapsed time(s)= 9.05 | diff Utot= 0.000000000016 | diff Force= 0.000000000019 |
| 9 | input_example/H2O-EF.dat | Elapsed time(s)= 3.04 | diff Utot= 0.000000000002 | diff Force= 0.000000000001 |
| 10 | input_example/H2O.dat | Elapsed time(s)= 2.69 | diff Utot= 0.000000000000 | diff Force= 0.000000000019 |
| 11 | input_example/HMn.dat | Elapsed time(s)= 10.50 | diff Utot= 0.000000000085 | diff Force= 0.000000000022 |
| 12 | input_example/Methane.dat | Elapsed time(s)= 2.05 | diff Utot= 0.000000000003 | diff Force= 0.000000000002 |
| 13 | input_example/Mol_MnO.dat | Elapsed time(s)= 6.41 | diff Utot= 0.000000000617 | diff Force= 0.000000000018 |
| 14 | input_example/Ndia2.dat | Elapsed time(s)= 5.59 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |

Total elapsed time (s) 177.21

System C (enaga) at ISSP, Univ. of Tokyo (Intel Xeon 6148 20core 2.4GHz)
icc version 18.0.5, compiler option: -O3 -xHOST -ip -no-prec-div -qopenmp -Dkcomp -fp-model precise
5 processes (MPI) x 4 thread (OpenMP)

| | | | | |
|---|---|---|---|---|
| 1 | input_example/Benzene.dat | Elapsed time(s)= 2.92 | diff Utot= 0.000000000025 | diff Force= 0.000000000002 |
| 2 | input_example/C60.dat | Elapsed time(s)= 9.47 | diff Utot= 0.000000000005 | diff Force= 0.000000000003 |
| 3 | input_example/CO.dat | Elapsed time(s)= 5.71 | diff Utot= 0.000000000072 | diff Force= 0.000000001573 |
| 4 | input_example/Cr2.dat | Elapsed time(s)= 5.46 | diff Utot= 0.000000000845 | diff Force= 0.000000000111 |
| 5 | input_example/Crys-MnO.dat | Elapsed time(s)= 40.05 | diff Utot= 0.000000000002 | diff Force= 0.000000000066 |
| 6 | input_example/GaAs.dat | Elapsed time(s)= 37.81 | diff Utot= 0.000000000009 | diff Force= 0.000000000001 |
| 7 | input_example/Glycine.dat | Elapsed time(s)= 2.96 | diff Utot= 0.000000000001 | diff Force= 0.000000000001 |
| 8 | input_example/Graphite4.dat | Elapsed time(s)= 5.76 | diff Utot= 0.000000000002 | diff Force= 0.000000000140 |
| 9 | input_example/H2O-EF.dat | Elapsed time(s)= 2.44 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |
| 10 | input_example/H2O.dat | Elapsed time(s)= 2.39 | diff Utot= 0.000000000002 | diff Force= 0.000000003224 |
| 11 | input_example/HMn.dat | Elapsed time(s)= 10.08 | diff Utot= 0.000000000129 | diff Force= 0.000000000020 |
| 12 | input_example/Methane.dat | Elapsed time(s)= 1.88 | diff Utot= 0.000000000001 | diff Force= 0.000000000000 |
| 13 | input_example/Mol_MnO.dat | Elapsed time(s)= 6.09 | diff Utot= 0.000000000272 | diff Force= 0.000000000150 |
| 14 | input_example/Ndia2.dat | Elapsed time(s)= 4.02 | diff Utot= 0.000000000000 | diff Force= 0.000000000001 |

Total elapsed time (s) 137.06

hster at JAIST (Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz)
icc compiler Ver. 14.0.2.144, compiler option: -openmp -O3 -xAVX -ip -no-prec-div
20 processes (MPI) x 1 thread (OpenMP)

| | | | | |
|---|---|---|---|---|
| 1 | input_example/Benzene.dat | Elapsed time(s)= 5.30 | diff Utot= 0.000000000038 | diff Force= 0.000000000003 |
| 2 | input_example/C60.dat | Elapsed time(s)= 12.53 | diff Utot= 0.000000000001 | diff Force= 0.000000000002 |
| 3 | input_example/CO.dat | Elapsed time(s)= 10.55 | diff Utot= 0.000000000047 | diff Force= 0.000000007948 |
| 4 | input_example/Cr2.dat | Elapsed time(s)= 10.74 | diff Utot= 0.000000000381 | diff Force= 0.000000000102 |
| 5 | input_example/Crys-MnO.dat | Elapsed time(s)= 27.48 | diff Utot= 0.000000000001 | diff Force= 0.000000000035 |
| 6 | input_example/GaAs.dat | Elapsed time(s)= 38.56 | diff Utot= 0.000000000001 | diff Force= 0.000000000001 |
| 7 | input_example/Glycine.dat | Elapsed time(s)= 5.76 | diff Utot= 0.000000000001 | diff Force= 0.000000000000 |
| 8 | input_example/Graphite4.dat | Elapsed time(s)= 6.73 | diff Utot= 0.000000000003 | diff Force= 0.000000000073 |
| 9 | input_example/H2O-EF.dat | Elapsed time(s)= 5.00 | diff Utot= 0.000000000001 | diff Force= 0.000000000001 |
| 10 | input_example/H2O.dat | Elapsed time(s)= 4.86 | diff Utot= 0.000000000000 | diff Force= 0.000000000020 |
| 11 | input_example/HMn.dat | Elapsed time(s)= 13.97 | diff Utot= 0.000000000118 | diff Force= 0.000000000001 |
| 12 | input_example/Methane.dat | Elapsed time(s)= 4.36 | diff Utot= 0.000000000006 | diff Force= 0.000000000002 |
| 13 | input_example/Mol_MnO.dat | Elapsed time(s)= 9.83 | diff Utot= 0.000000000144 | diff Force= 0.000000000079 |
| 14 | input_example/Ndia2.dat | Elapsed time(s)= 8.39 | diff Utot= 0.000000000000 | diff Force= 0.000000000001 |

Total elapsed time (s) 164.04

CRAY-XC40 at JAIST (Intel Xeon E5-2695v4 2.1GHz)
icc version 17.0.7, compiler option: -Dxt3 -O3 -axCOMMON-AVX512,CORE-AVX512,CORE-AVX2,CORE-AVX-I,AVX,SSE4.2,SSE4.1,SSE3,SSSE3,SSE2 -qopenmp
18 processes (MPI) x 2 thread (OpenMP)

| | | | | |
|---|---|---|---|---|
| 1 | input_example/Benzene.dat | Elapsed time(s)= 4.23 | diff Utot= 0.000000000040 | diff Force= 0.000000000002 |
| 2 | input_example/C60.dat | Elapsed time(s)= 12.40 | diff Utot= 0.000000000001 | diff Force= 0.000000000001 |
| 3 | input_example/CO.dat | Elapsed time(s)= 9.09 | diff Utot= 0.000000000150 | diff Force= 0.000000009551 |
| 4 | input_example/Cr2.dat | Elapsed time(s)= 8.56 | diff Utot= 0.000000000462 | diff Force= 0.000000000004 |
| 5 | input_example/Crys-MnO.dat | Elapsed time(s)= 20.81 | diff Utot= 0.000000000001 | diff Force= 0.000000000014 |
| 6 | input_example/GaAs.dat | Elapsed time(s)= 31.99 | diff Utot= 0.000000000001 | diff Force= 0.000000000001 |
| 7 | input_example/Glycine.dat | Elapsed time(s)= 4.71 | diff Utot= 0.000000000001 | diff Force= 0.000000000002 |
| 8 | input_example/Graphite4.dat | Elapsed time(s)= 4.89 | diff Utot= 0.000000000032 | diff Force= 0.000000000004 |
| 9 | input_example/H2O-EF.dat | Elapsed time(s)= 4.03 | diff Utot= 0.000000000001 | diff Force= 0.000000000002 |
| 10 | input_example/H2O.dat | Elapsed time(s)= 3.83 | diff Utot= 0.000000000001 | diff Force= 0.000000001042 |
| 11 | input_example/HMn.dat | Elapsed time(s)= 12.73 | diff Utot= 0.000000000064 | diff Force= 0.000000000029 |
| 12 | input_example/Methane.dat | Elapsed time(s)= 3.24 | diff Utot= 0.000000000004 | diff Force= 0.000000000001 |
| 13 | input_example/Mol_MnO.dat | Elapsed time(s)= 8.32 | diff Utot= 0.000000000576 | diff Force= 0.000000000032 |
| 14 | input_example/Ndia2.dat | Elapsed time(s)= 6.12 | diff Utot= 0.000000000000 | diff Force= 0.000000000001 |

Total elapsed time (s) 134.96

FX100 at Nagoya Univ. (PRIMEHPC FX100, SPARC64b XIfx, 2.2Gz)
mpifccpx, compiler option: -Kfast -Kopenmp -Dnosse -Dkcomp
16 processes (MPI) x 2 thread (OpenMP)

| | | | | |
|---|---|---|---|---|
| 1 | input_example/Benzene.dat | Elapsed time(s)= 12.29 | diff Utot= 0.000000000003 | diff Force= 0.000000000005 |
| 2 | input_example/C60.dat | Elapsed time(s)= 29.15 | diff Utot= 0.000000000158 | diff Force= 0.000000000050 |
| 3 | input_example/CO.dat | Elapsed time(s)= 38.71 | diff Utot= 0.000000000125 | diff Force= 0.000000003104 |
| 4 | input_example/Cr2.dat | Elapsed time(s)= 25.43 | diff Utot= 0.000000001020 | diff Force= 0.000000000007 |
| 5 | input_example/Crys-MnO.dat | Elapsed time(s)= 84.38 | diff Utot= 0.000000006058 | diff Force= 0.000000073199 |
| 6 | input_example/GaAs.dat | Elapsed time(s)= 80.54 | diff Utot= 0.000000000011 | diff Force= 0.000000015689 |
| 7 | input_example/Glycine.dat | Elapsed time(s)= 14.78 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |
| 8 | input_example/Graphite4.dat | Elapsed time(s)= 14.52 | diff Utot= 0.000000000016 | diff Force= 0.000000000001 |
| 9 | input_example/H2O-EF.dat | Elapsed time(s)= 12.78 | diff Utot= 0.000000000001 | diff Force= 0.000000000001 |
| 10 | input_example/H2O.dat | Elapsed time(s)= 13.07 | diff Utot= 0.000000000001 | diff Force= 0.000000000023 |
| 11 | input_example/HMn.dat | Elapsed time(s)= 37.63 | diff Utot= 0.000000000153 | diff Force= 0.000000000000 |
| 12 | input_example/Methane.dat | Elapsed time(s)= 9.82 | diff Utot= 0.000000000007 | diff Force= 0.000000000002 |
| 13 | input_example/Mol_MnO.dat | Elapsed time(s)= 26.48 | diff Utot= 0.000000000209 | diff Force= 0.000000000058 |
| 14 | input_example/Ndia2.dat | Elapsed time(s)= 19.74 | diff Utot= 0.000000000000 | diff Force= 0.000000000001 |

Total elapsed time (s) 419.32

pauli (AMD EPYC 7351P, 2.4GHz), a machine in the Ozaki laboratory
gcc version 7.4.0, compiler option: -Dkcomp -O3 -march=znver1 -mtune=znver1 -mfma -mavx2 -m3dnow -fomit-frame-pointer -fopenmp
12 processes (MPI) x 1 thread (OpenMP)

| | | | | |
|---|---|---|---|---|
| 1 | input_example/Benzene.dat | Elapsed time(s)= 3.32 | diff Utot= 0.000000000039 | diff Force= 0.000000000002 |
| 2 | input_example/C60.dat | Elapsed time(s)= 13.49 | diff Utot= 0.000000000013 | diff Force= 0.000000000006 |
| 3 | input_example/CO.dat | Elapsed time(s)= 9.13 | diff Utot= 0.000000000064 | diff Force= 0.000000000934 |
| 4 | input_example/Cr2.dat | Elapsed time(s)= 8.43 | diff Utot= 0.000000002324 | diff Force= 0.000000000157 |
| 5 | input_example/Crys-MnO.dat | Elapsed time(s)= 25.40 | diff Utot= 0.000000000003 | diff Force= 0.000000000070 |
| 6 | input_example/GaAs.dat | Elapsed time(s)= 38.09 | diff Utot= 0.000000000002 | diff Force= 0.000000000001 |
| 7 | input_example/Glycine.dat | Elapsed time(s)= 4.41 | diff Utot= 0.000000000001 | diff Force= 0.000000000003 |
| 8 | input_example/Graphite4.dat | Elapsed time(s)= 4.83 | diff Utot= 0.000000000015 | diff Force= 0.000000000011 |
| 9 | input_example/H2O-EF.dat | Elapsed time(s)= 3.55 | diff Utot= 0.000000000000 | diff Force= 0.000000000001 |
| 10 | input_example/H2O.dat | Elapsed time(s)= 2.95 | diff Utot= 0.000000000001 | diff Force= 0.000000000806 |
| 11 | input_example/HMn.dat | Elapsed time(s)= 11.85 | diff Utot= 0.000000000113 | diff Force= 0.000000000001 |
| 12 | input_example/Methane.dat | Elapsed time(s)= 2.72 | diff Utot= 0.000000000006 | diff Force= 0.000000000001 |
| 13 | input_example/Mol_MnO.dat | Elapsed time(s)= 8.01 | diff Utot= 0.000000000326 | diff Force= 0.000000000050 |
| 14 | input_example/Ndia2.dat | Elapsed time(s)= 5.93 | diff Utot= 0.000000000000 | diff Force= 0.000000000000 |

Total elapsed time (s) 142.11

# 6 Automatic running test with large-scale systems

In some cases, one may want to know machine performance for more time consuming calculations. For this purpose, an automatic running test with relatively large-scale systems can be performed by

### For the MPI parallel running

```
% mpirun -np 112 openmx -runtestL
```

### For the MPI/OpenMP parallel running

```
% mpirun -np 112 openmx -runtestL -nt 2
```

Then, OpenMX will run with 16 test files, and compare calculated results with the reference results which are stored in 'work/large_example'. The comparison (absolute difference in the total energy and force) is stored in a file 'runtestL.result' in the directory 'work'. The reference results were calculated using 28 MPI processes of a 2.6 GHz Xeon cluster machine. If the difference is within last seven digits, we may consider that the installation is successful. As an example, 'runtestL.result' generated by the automatic running test is shown below:

| | | | | |
|---|---|---|---|---|
| 1 | large_example/5_5_13COb2.dat | Elapsed time(s)= 52.78 | diff Utot= 0.000000000020 | diff Force= 0.000000000004 |
| 2 | large_example/B2C62_Band.dat | Elapsed time(s)= 403.51 | diff Utot= 0.000000000001 | diff Force= 0.000000063810 |
| 3 | large_example/CG15c-DC-LNO.dat | Elapsed time(s)= 103.31 | diff Utot= 0.000000000269 | diff Force= 0.000000000551 |
| 4 | large_example/DIA512-1.dat | Elapsed time(s)= 49.35 | diff Utot= 0.000000027379 | diff Force= 0.000000031436 |
| 5 | large_example/FeBCC.dat | Elapsed time(s)= 80.54 | diff Utot= 0.000000000016 | diff Force= 0.000000000001 |
| 6 | large_example/GEL.dat | Elapsed time(s)= 44.95 | diff Utot= 0.000000000009 | diff Force= 0.000000000004 |
| 7 | large_example/GFRAG.dat | Elapsed time(s)= 27.68 | diff Utot= 0.000000000001 | diff Force= 0.000000000001 |
| 8 | large_example/GGFF.dat | Elapsed time(s)= 643.36 | diff Utot= 0.000000000037 | diff Force= 0.000000000809 |
| 9 | large_example/MCCN.dat | Elapsed time(s)= 82.04 | diff Utot= 0.000000005885 | diff Force= 0.000000003486 |
| 10 | large_example/Mn12_148_F.dat | Elapsed time(s)= 74.25 | diff Utot= 0.000000000015 | diff Force= 0.000000000010 |
| 11 | large_example/N1C999.dat | Elapsed time(s)= 1212.42 | diff Utot= 0.000000000035 | diff Force= 0.000000000390 |
| 12 | large_example/Ni63-O64.dat | Elapsed time(s)= 70.90 | diff Utot= 0.000000000211 | diff Force= 0.000000000008 |
| 13 | large_example/Pt63.dat | Elapsed time(s)= 58.76 | diff Utot= 0.000000001297 | diff Force= 0.000000000242 |
| 14 | large_example/SialicAcid.dat | Elapsed time(s)= 16.75 | diff Utot= 0.000000000001 | diff Force= 0.000000000001 |
| 15 | large_example/ZrB2_2x2.dat | Elapsed time(s)= 133.10 | diff Utot= 0.000000000044 | diff Force= 0.000000000020 |
| 16 | large_example/nsV4Bz5.dat | Elapsed time(s)= 99.37 | diff Utot= 0.000000004771 | diff Force= 0.000000003167 |

Total elapsed time (s) 3153.08

The comparison was made using 112 MPI processes on the same Xeon cluster machine. Since the automatic running test requires large memory, you may encounter a segmentation fault in case that a small number of cores are used. Also the above example implies that the total elapsed time is about 53 minutes even using 112 cores. See also the Section 'Large-scale calculation' for another large-scale benchmark calculation.

# 7 Input file

## 7.1 An example: methane molecule

An input file 'Methane.dat' in the directory 'work' is shown below. The input file has a flexible data format in such a way that a parameter is given behind a keyword, the order of keywords is arbitrary, and a blank and a comment can be also described freely. For the keywords and options, both capital, small letters, and the mixture are acceptable, although these options in below example are written in a specific form.

```
#
# File Name
#

System.CurrrentDirectory         ./    # default=./
System.Name                      met
level.of.stdout                   1    # default=1 (1-3)
level.of.fileout                  1    # default=1 (0-2)

#
# Definition of Atomic Species
#

Species.Number       2
<Definition.of.Atomic.Species
 H   H5.0-s1          H_PBE19
 C   C5.0-s1p1        C_PBE19
Definition.of.Atomic.Species>

#
# Atoms
#

Atoms.Number         5
Atoms.SpeciesAndCoordinates.Unit   Ang # Ang|AU
<Atoms.SpeciesAndCoordinates
 1  C      0.000000    0.000000    0.000000     2.0  2.0
 2  H     -0.889981   -0.629312    0.000000     0.5  0.5
 3  H      0.000000    0.629312   -0.889981     0.5  0.5
 4  H      0.000000    0.629312    0.889981     0.5  0.5
 5  H      0.889981   -0.629312    0.000000     0.5  0.5
Atoms.SpeciesAndCoordinates>
Atoms.UnitVectors.Unit             Ang # Ang|AU
<Atoms.UnitVectors
  10.0   0.0   0.0
   0.0  10.0   0.0
   0.0   0.0  10.0
Atoms.UnitVectors>

#
```

```
# SCF or Electronic System
#

scf.XcType                  GGA-PBE     # LDA|LSDA-CA|LSDA-PW|GGA-PBE
scf.SpinPolarization         off        # On|Off|NC
scf.ElectronicTemperature   300.0       # default=300 (K)
scf.energycutoff            120.0       # default=150 (Ry)
scf.maxIter                  100        # default=40
scf.EigenvalueSolver         cluster    # DC|Cluster|Band
scf.Kgrid                   1 1 1       # means n1 x n2 x n3
scf.Mixing.Type            rmm-diis     # Simple|Rmm-Diis|Gr-Pulay|Kerker|Rmm-Diisk
scf.Init.Mixing.Weight      0.30        # default=0.30
scf.Min.Mixing.Weight       0.001       # default=0.001
scf.Max.Mixing.Weight       0.400       # default=0.40
scf.Mixing.History           7          # default=5
scf.Mixing.StartPulay        5          # default=6
scf.criterion              1.0e-10      # default=1.0e-6 (Hartree)

#
# MD or Geometry Optimization
#

MD.Type                       nomd       # Nomd|Opt|NVE|NVT_VS|NVT_NH
                                       # Constraint_Opt|DIIS
MD.maxIter                      1        # default=1
MD.TimeStep                    1.0       # default=0.5 (fs)
MD.Opt.criterion           1.0e-4        # default=1.0e-4 (Hartree/Bohr)
```

### 7.2 Keywords

The specification of each keyword is given below. The list does not include all the keywords in OpenMX, and those keywords will be explained in each corresponding section.

## File name

**System.CurrrentDir**
The output directory of output files is specified by this keyword. The default is './'.

**System.Name**
The file name of output files is specified by this keyword.

**DATA.PATH**
The path to the VPS and PAO directories can be specified in your input file by the following keyword:

```
DATA.PATH    ../DFT_DATA19     # default=../DFT_DATA19
```

Both the absolute and relative specifications are available. The default is '../DFT_DATA19'.

**level.of.stdout**
The amount of the standard output during the calculation is controlled by the keyword 'level.of.stdout'. In case of 'level.of.stdout=0', minimum information. In case of 'level.of.stdout=1', standard information. In case of 'level.of.stdout=2', additional information together with the minimum output information. 'level.of.stdout=3' is for developers. The default is 1.

**level.of.fileout**
The amount of information output to the files is controlled by the keyword 'level.of.fileout'. In case of 'level.of.fileout=0', minimum information (no Gaussian cube and grid files). In case of 'level.of.fileout=1', standard output. In case of 'level.of.fileout=2', additional information together with the standard output. The default is 1.

## Definition of Atomic Species

**Species.Number**
The number of atomic species in the system is specified by the keyword 'Species.Number'.

**Definition.of.Atomic.Species**
Please specify atomic species by giving both the file name of pseudo-atomic basis orbitals and pseudopotentials which must be existing in the directories 'DFT_DATA19/PAO' and 'DFT_DATA19/VPS', respectively. For example, they are specified as follows:

```
<Definition.of.Atomic.Species
 H   H5.0-s1>1p1>1      H_CA19
 C   C5.0-s1>1p1>1      C_CA19
Definition.of.Atomic.Species>
```

The beginning of the description must be '<Definition.of.Atomic.Species', and the last of the description must be 'Definition.of.Atomic.Species>'. In the first column, you can give any name to specify the atomic species. The name is used in the specification of atomic coordinates by 'Atoms.SpeciesAndCoordinates'. In the second column, the file name of the pseudo-atomic basis orbitals without the file extension and the number of primitive orbitals and contracted orbitals are given. Here we introduce an abbreviation of the basis orbital we used as H4.0-s1>1p1>1, where H4.0 indicates the file name of the pseudo-atomic basis orbitals without the file extension which must exist in the directory 'DFT_DATA19/PAO', s1>1 means that one optimized orbitals are constructed from one primitive orbitals for the s-orbital, which means no contraction. Also, in case of s1>1, corresponding to no contraction, you can use a simple notation 's1' instead of 's1>1'. Thus, 'H4.0-s1p1' is equivalent to 'H4.0-s1>1p1>1'. In the third column, the file name for the pseudopotentials without the file extension is given. Also the file must exist in the directory 'DFT_DATA19/VPS'. It can be possible to assign as the different atomic species for the same atomic element by specifying the different basis orbitals and pseudopotentials. For example, you can define the atomic species as follows:

```
<Definition.of.Atomic.Species
 H1   H5.0-s1p1         H_CA19
 H2   H5.0-s2p2d1       H_CA19
```

```
 C1   C5.0-s2p2         C_CA19
 C2   C5.0-s2p2d2       C_CA19
Definition.of.Atomic.Species>
```

The flexible definition may be useful for the decrease of computational efforts, in which only high level basis functions are used for atoms belonging to the essential part which determines the electric properties in the system, and lower level basis functions are used for atoms in the other inert parts.

## Atoms

**Atoms.Number**
The total number of atoms in the system is specified by the keyword 'Atoms.Number'.

**Atoms.SpeciesAndCoordinates.Unit**
The unit of the atomic coordinates is specified by the keyword 'Atoms.SpeciesAndCoordinates.Unit'. Please specify 'Ang' when you use the unit of Angstrom, and 'AU' when the unit of atomic unit. The fractional coordinate is also available by 'FRAC'. Then, please specify the coordinates spanned by **a**, **b**, and **c**-axes given in 'Atoms.UnitVectors'. In the fractional coordinates, the coordinates can range from 0.0 to 1.0, and the coordinates beyond its range will be automatically adjusted after the input file is read.

**Atoms.SpeciesAndCoordinates**
The atomic coordinates and the number of spin charge are given by the keyword 'Atoms.SpeciesAndCoordinates' as follows:

```
<Atoms.SpeciesAndCoordinates
  1   C      0.000000    0.000000    0.000000     2.0  2.0
  2   H     -0.889981   -0.629312    0.000000     0.5  0.5
  3   H      0.000000    0.629312   -0.889981     0.5  0.5
  4   H      0.000000    0.629312    0.889981     0.5  0.5
  5   H      0.889981   -0.629312    0.000000     0.5  0.5
Atoms.SpeciesAndCoordinates>
```

The beginning of the description must be '<Atoms.SpeciesAndCoordinates', and the last of the description must be 'Atoms.SpeciesAndCoordinates>'. The first column is a sequential serial number for identifying atoms. The second column is given to specify the atomic species which must be given in the first column of the specification of the keyword 'Definition.of.Atomic.Species' in advance. In the third, fourth, and fifth columns, $x$-, $y$-, and $z$-coordinates are given. When 'FRAC' is chosen for the keyword 'Atoms.SpeciesAndCoordinates.Unit', the third, fourth, and fifth columns are fractional coordinates spanned by **a**, **b**, and **c**-axes, where the coordinates can range from 0.0 to 1.0, and the coordinates beyond its range will be automatically adjusted after the input file is read. The sixth and seventh columns give the number of initial charges for up and down spin states of each atom, respectively. The sum of up and down charges must be the number of valence electrons for the atomic element. When you calculate spin-polarized systems using 'LSDA-CA' or 'LSDA-PW', you can give

the initial spin charges for each atom, which might be those of the ground state, to accelerate the SCF convergence.

**Atoms.UnitVectors.Unit**
The unit of the vectors for the unit cell is specified by the keyword 'Atoms.UnitVectors.Unit'. Please specify 'Ang' when you use the unit of Angstrom, and 'AU' when the unit of atomic unit.

**Atoms.UnitVectors**
The vectors, **a**, **b**, and **c** of the unit cell are given by the keyword 'Atoms.UnitVectors' as follows:

```
<Atoms.UnitVectors
  10.0   0.0   0.0
   0.0  10.0   0.0
   0.0   0.0  10.0
Atoms.UnitVectors>
```

The beginning of the description must be '<Atoms.UnitVectors', and the last of the description must be 'Atoms.UnitVectors>'. The first, second, and third rows correspond to the vectors, **a**, **b**, and **c** of the unit cell, respectively. If the keyword is absent in the cluster calculation, a unit cell is automatically determined so that the isolated system cannot overlap with the image systems in the repeated cells via basis functions. See also the Section 'Automatic determination of the cell size'.

## SCF or Electronic System

**scf.XcType**
The keyword 'scf.XcType' specifies the exchange-correlation potential. Currently, 'LDA', 'LSDA-CA', 'LSDA-PW', and 'GGA-PBE' are available, where 'LSDA-CA' is the local spin density functional of Ceperley-Alder [2], 'LSDA-PW' is the local spin density functional of Perdew-Wang, in which the gradient of density is set to zero in their GGA formalism [4]. Note: 'LSDA-CA' is faster than 'LSDA-PW'. 'GGA-PBE' is a GGA functional proposed by Perdew et al [5].

**scf.SpinPolarization**
The keyword 'scf.SpinPolarization' specifies the non-spin polarization or the spin polarization for the electronic structure. If the calculation for the spin polarization is performed, then specify 'ON'. If the calculation for the non-spin polarization is performed, then specify 'OFF'. When you use 'LDA' for the keyword 'scf.XcType', the keyword 'scf.SpinPolarization' must be 'OFF'. In addition to these options, 'NC' is supported for the non-collinear DFT calculation. For this calculation, see also the Section 'Non-collinear DFT'.

**scf.partialCoreCorrection**
The keyword 'scf.partialCoreCorrection' is a flag for a partial core correction (PCC) in calculations of exchange-correlation energy and potential. 'ON' means that PCC is made, and 'OFF' is none. In any cases, the flag should be 'ON', since pseudopotentials generated with PCC should be used with PCC, and also PCC does not affect the result for pseudopotentials without PCC because of zero PCC charge in this case.

**scf.Hubbard.U**
In case of the LDA+U or GGA+U calculation, the keyword 'scf.Hubbard.U' should be switched 'ON' (ON|OFF). The default is 'OFF'.

**scf.Hubbard.Occupation**
In the LDA+U method, three occupation number operators 'onsite', 'full', and 'dual' are available which can be specified by the keyword 'scf.Hubbard.Occupation'.

**Hubbard.U.values**
An effective U-value on each orbital of species is defined by the following keyword:

```
<Hubbard.U.values                  #  eV
 Ni  1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 4.0 2d 0.0
 O   1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 0.0
Hubbard.U.values>
```

The beginning of the description must be '<Hubbard.U.values', and the last of the description must be 'Hubbard.U.values>'. For all the basis orbitals specified by the 'Definition.of.Atomic.Species', you have to give an effective U-value in the above format. The '1s' and '2s' mean the first and second $s$-orbital, and the number behind '1s' is the effective U-value (eV) for the first $s$-orbital. The same rule is applied to $p$- and $d$-orbitals.

**scf.Constraint.NC.Spin**
The keyword 'scf.Constraint.NC.Spin' should be switched 'ON' (ON|OFF) when the constraint DFT method for the non-collinear spin orientation is performed.

**scf.Constraint.NC.Spin.v**
The keyword 'scf.Constraint.NC.Spin.v' gives a prefactor (eV) of the penalty functional in the constraint DFT for the non-collinear spin orientation.

**scf.ElectronicTemperature**
The electronic temperature (K) is given by the keyword 'scf.ElectronicTemperature'. The default is 300 (K).

**scf.energycutoff**
The keyword 'scf.energycutoff' specifies the cutoff energy which is used in the calculation of matrix elements associated with difference charge Coulomb potential and exchange-correlation potential and the solution of Poisson's equation using fast Fourier transform (FFT). The default is 150 (Ryd).

**scf.Ngrid**
The keyword 'scf.Ngrid' gives the number of grids to discretize the **a**-, **b**-, and **c**-axes. Although 'scf.energycutoff' is usually used for the discretization, when the numbers of grids are specified by 'scf.Ngrid', they are used for the discretization instead of those by 'scf.energycutoff'.

**scf.maxIter**
The maximum number of SCF iterations is specified by the keyword 'scf.maxIter'. The SCF loop is terminated at the number specified by 'scf.maxIter' even if a convergence criterion is not satisfied. The default is 40.

**scf.EigenvalueSolver**
The solution method for the eigenvalue problem is specified by the keyword 'scf.EigenvalueSolver'. An O($N$) divide-conquer method 'DC', an O($N$) divide-conquer method with localized natural orbitals 'DC-LNO', an O($N$) Krylov subspace method 'Krylov', a numerically exact low-order scaling method 'ON2', the cluster calculation 'Cluster', and the band calculation 'Band' are available.

**scf.Kgrid**
When you specify the band calculation 'Band' for the keyword 'scf.EigenvalueSolver', then you need to give a set of numbers (n1,n2,n3) of grids to discretize the first Brillouin zone in the k-space by the keyword 'scf.Kgrid'. For the reciprocal vectors $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$, and $\tilde{\mathbf{c}}$ in the k-space, please provide a set of numbers (n1,n2,n3) of grids as n1 n2 n3. The k-points in OpenMX are generated by a regular mesh method.

**scf.ProExpn.VNA**
Switch on the keyword 'scf.ProExpn.VNA' in case that the neutral atom potential VNA is expanded by projector operators [42]. Otherwise turn off. The default is 'ON'.

```
scf.ProExpn.VNA      ON      # ON|OFF, default = ON
```

In case that 'scf.ProExpn.VNA=OFF', the matrix elements for the VNA potential are evaluated by using the regular mesh in real space.

**scf.Mixing.Type**
A mixing method of the electron density (or the density matrix) to generate an input electron density at the next SCF step is specified by keyword 'scf.Mixing.Type'. A simple mixing method ('Simple'), 'GR-Pulay' method (Guaranteed-Reduction Pulay method) [57], 'RMM-DIIS' method [58], 'Kerker' method [59], 'RMM-DIISK' method [58], 'RMM-DIISV' method [58], and 'RMM-DIISH' method [58] are available. The simple mixing method used here is modified to accelerate the convergence, referring to a convergence history. When 'GR-Pulay', 'RMM-DIIS', 'Kerker', 'RMM-DIISK', 'RMM-DIISV', or 'RMM-DIISH' is used, the following recipes are helpful to obtain faster convergence of SCF calculations:

- Use a rather larger value for 'scf.Mixing.StartPulay'. Before starting the Pulay-like mixing, achieve a convergence at some level. An appropriate value may be 10 to 30 for 'scf.Mixing.StartPulay'.
- Use a rather larger value for 'scf.ElectronicTemperature' in case of metallic systems. When 'scf.ElectronicTemperature' is too low, numerical instabilities appear often.
- Use a large value for 'scf.Mixing.History'. In most cases, 'scf.Mixing.History=30' can be a good value.

Among these mixing schemes, the robustest one might be 'RMM-DIISK'.

**scf.Init.Mixing.Weight**
The keyword 'scf.Init.Mixing.Weight' gives the initial mixing weight used by the simple mixing, the GR-Pulay, the RMM-DIIS, the Kerker, the RMM-DIISK, the RMM-DIISV, and the RMM-DIISH methods. The valid range is 0 <scf.Init.Mixing.Weight< 1. The default is 0.3.

**scf.Min.Mixing.Weight**

The keyword 'scf.Min.Mixing.Weight' gives the lower limit of a mixing weight in the simple and Kerker mixing methods. The default is 0.001.

**scf.Max.Mixing.Weight**
The keyword 'scf.Max.Mixing.Weight' gives the upper limit of a mixing weight in the simple and Kerker mixing methods. The default is 0.4.

**scf.Kerker.factor**
The keyword gives a Kerker factor which is used in the Kerker and RMM-DIISK mixing methods. If the keyword is not given, a proper value is automatically determined. For further details, see the Section 'SCF convergence'.

**scf.Mixing.History**
In the GR-Pulay method [57], the RMM-DIIS method [58], the Kerker method [59], the RMM-DIISK method [58], the RMM-DIISV method [58], and the RMM-DIISH method [58], the input electron density (Hamiltonian) at the next SCF step is estimated based on the output electron densities (Hamiltonian) in the several previous SCF steps. The keyword 'scf.Mixing.History' specifies the number of previous SCF steps which are used in the estimation. For example, if 'scf.Mixing.History' is specified to be 3, and when the SCF step is 6th, the electron densities at 5, 4, and 3 SCF steps are taken into account. Around 30 is a better choice.

**scf.Mixing.StartPulay**
The SCF step which starts the GR-Pulay, the RMM-DIIS, the Kerker, the RMM-DIISK, the RMM-DIISV method, or the RMM-DIISH methods is specified by the keyword 'scf.Mixing.StartPulay'. The SCF steps before starting these Pulay-type methods are then performed by the simple or Kerker mixing methods. The default is 6.

**scf.Mixing.EveryPulay**
The residual vectors in the Pulay-type mixing methods tend to become linearly dependent each other as the mixing steps accumulate, and the linear dependence among the residual vectors makes the convergence difficult. A way of avoiding the linear dependence is to do the Pulay-type mixing occasionally during the Kerker mixing. With this prescription, you can specify the frequency using the keyword 'scf.Mixing.EveryPulay'. For example, in case of 'scf.Mixing.EveryPulay=5', the Pulay-mixing is made at every five SCF iterations, while the Kerker mixing is used at the other steps. 'scf.Mixing.EveryPulay=1' corresponds to the conventional Pulay-type mixing. It is noted that the keyword 'scf.Mixing.EveryPulay' is supported for only 'RMM-DIISK', and the default value is 1.

**scf.criterion**
The keyword 'scf.criterion' specifies a convergence criterion (Hartree) for the SCF calculation. The SCF iteration is ended when a condition, dUele<scf.criterion, is satisfied, where dUele is defined as the absolute deviation between the eigenvalue energy at the current and previous SCF steps. The default is 1.0e-6 (Hartree).

**scf.Electric.Field**
The keyword 'scf.Electric.Field' gives the strength of a uniform external electric field given by a sawtooth waveform. For example, when an electric field of 1.0 GV/m ($10^9$ V/m) is applied along the **a**-axis, specify in your input file as follows:

```
scf.Electric.Field   1.0 0.0 0.0   # default=0.0 0.0 0.0 (GV/m)
```

The sign of electric field is taken as that applied to electrons. The default is 0.0 0.0 0.0.

**scf.system.charge**
The keyword 'scf.system.charge' gives the amount of the electron and hole dopings. The plus and minus signs correspond to hole and electron dopings, respectively. The default is 0.

**scf.SpinOrbit.Coupling**
When the spin-orbit coupling is included, the keyword should be 'ON', otherwise please set to 'OFF'. In case of the inclusion of the spin-orbit coupling, you have to use j-dependent pseudopotentials. See also the Section 'Relativistic effects' as for the j-dependent pseudopotentials.

## 1D FFT

**1DFFT.EnergyCutoff**
The keyword '1DFFT.EnergyCutoff' gives the energy range to tabulate the Fourier transformed radial functions of pseudo-atomic orbitals and of the projectors for non-local potentials. The default is 3600 (Ryd).

**1DFFT.NumGridK**
The keyword '1DFFT.NumGridK' gives the the number of radial grids in the k-space. The values of the Fourier transformation for radial functions of pseudo-atomic orbitals and of the projectors for non-local potentials are tabulated on the grids, ranging from zero to 1DFFT.EnergyCutoff, as a function of radial axis in the **k**-space. The default is 900.

**1DFFT.NumGridR**
The keyword '1DFFT.NumGridR' gives the the number of radial grids in real space which is used in the numerical grid integrations of the Fourier transformation for radial functions of pseudo-atomic orbitals and of the projectors for non-local potentials. The default is 900.

## Orbital Optimization

**orbitalOpt.Method**
The keyword 'orbitalOpt.Method' specifies a method for the orbital optimization. When the orbital optimization is not performed, then choose 'OFF'. When the orbital optimization is performed, the following two options are available: 'atoms' in which basis orbitals on each atom are fully optimized, 'species' in which basis orbitals on each species are optimized. In 'atoms', the radial functions of basis orbitals are optimized with a constraint that the radial wave function $R$ is independent on the magnetic quantum number, which guarantees the rotational invariance of the total energy. However, the optimized orbital on all the atoms can be different from each other. In the 'species', basis orbitals in atoms with the same species name, that you define in 'Definition.of.Atomic.Species', are optimized as the same orbitals. If you want to assign the same orbitals to atoms with almost the same chemical environment, and optimize these orbitals, this scheme is useful.

**orbitalOpt.scf.maxIter**
The maximum number of SCF iterations in the orbital optimization is specified by the keyword 'orbitalOpt.scf.maxIter'.

**orbitalOpt.Opt.maxIter**
The maximum number of iterations for the orbital optimization is specified by the keyword 'orbitalOpt.Opt.maxIter'. The iteration loop for the orbital optimization is terminated at the number specified by 'orbitalOpt.Opt.maxIter' even if a convergence criterion is not satisfied.

**orbitalOpt.Opt.Method**
Two schemes for the optimization of orbitals are available: 'EF' which is an eigenvector following method, 'DIIS' which is the direct inversion method in iterative subspace. The algorithms are basically the same as for the geometry optimization. Either 'EF' or 'DIIS' is chosen by the keyword 'orbitalOpt.Opt.Method'.

**orbitalOpt.StartPulay**
The quasi Newton method 'EF' and 'DIIS' starts from the optimization step specified by the keyword 'orbitalOpt.StartPulay'.

**orbitalOpt.HistoryPulay**
The keyword 'orbitalOpt.HistoryPulay' specifies the number of previous steps to estimate the next input contraction coefficients used in the quasi Newton method 'EF' and 'DIIS'.

**orbitalOpt.SD.step**
The orbital optimization at optimization steps before moving to the quasi Newton method 'EF' or 'DIIS' is performed by the steepest decent method. The prefactor used in the steepest decent method is specified by the keyword 'orbitalOpt.SD.step'. In most cases, 'orbitalOpt.SD.step' of 0.001 can be a good prefactor.

**orbitalOpt.criterion**
The keyword 'orbitalOpt.criterion' specifies a convergence criterion ((Hartree/Borg)$^2$) for the orbital optimization. The iterations loop is finished when a condition, Norm of derivatives<orbitalOpt.criterion, is satisfied.

**CntOrb.fileout**
If you want to output the optimized radial orbitals to files, then the keyword 'CntOrb.fileout' must be 'ON'.

**Num.CntOrb.Atoms**
The keyword 'Num.CntOrb.Atoms' gives the number of atoms whose optimized radial orbitals are output to files.

**Atoms.Cont.Orbitals**
The keyword 'Atoms.Cont.Orbitals' specifies the atom number, which is given by the first column in the specification of the keyword 'Atoms.SpeciesAndCoordinates' for the output of optimized orbitals as follows:

```
<Atoms.Cont.Orbitals
 1
```

```
 2
Atoms.Cont.Orbitals>
```

The beginning of the description must be '<Atoms.Cont.Orbitals', and the last of the description must be 'Atoms.Cont.Orbitals>'. The number of lines should be consistent with the number specified in the keyword 'Atoms.Cont.Orbitals'. For example, the name of files are C_1.pao and H_2.pao, where the symbol corresponds to that given by the first column in the specification of the keyword 'Definition.of.Atomic.Species' and the number after the symbol means that of the first column in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. These output files 'C_1.pao' and 'H_2.pao' can be an input data for pseudo-atomic orbitals as is.

## SCF Order-N

**orderN.HoppingRanges**
The keyword 'orderN.HoppingRanges' defines the radius of a sphere which is centered on each atom. The physically truncated cluster for each atom is constructed by picking up atoms inside the sphere with the radius in the DC, DC-LNO, and Krylov subspace O($N$) methods.

**orderN.KrylovH.order**
The dimension of the Krylov subspace of Hamiltonian in each truncated cluster is given by the 'orderN.KrylovH.order'.

**orderN.KrylovS.order**
In case of 'orderN.Exact.Inverse.S=off', the inverse is approximated by a Krylov subspace method for the inverse, where the dimension of the Krylov subspace of overlap matrix in each truncated cluster is given by the keyword 'orderN.KrylovS.order'. The default value is 'orderN.KrylovH.order'×4.

**orderN.Exact.Inverse.S**
In case of 'orderN.Exact.Inverse.S=on', the inverse of overlap matrix for each truncated cluster is exactly evaluated. Otherwise, see the keyword 'orderN.KrylovS.order'. The default is 'on' (on|off).

**orderN.Recalc.Buffer**
In case of 'orderN.Recalc.Buffer=on', the buffer matrix is recalculated at every SCF step. Otherwise, the buffer matrix is calculated at the first SCF step, and fixed at subsequent SCF steps. The default is 'on' (on|off).

**orderN.Expand.Core**
In case of 'orderN.Expand.Core=on', the core region is defined by atoms within a sphere with radius of $1.2 \times r_{\min}$, where $r_{\min}$ is the distance between the central atom and the nearest atom. The core region defines a set of vectors used for the first step in the generation of the Krylov subspace for each truncated cluster. In case of 'orderN.Expand.Core=off', the central atom is considered as the core region. The default is 'on' (on|off).

## MD or Geometry Optimization

**MD.Type**
Please specify the type of the molecular dynamics calculation or the geometry optimization. Currently, NO MD (Nomd), MD with the NVE ensemble (NVE), MD with the NVT ensemble by a velocity scaling scheme (NVT_VS)[30], MD with the NVT ensemble by a Nose-Hoover scheme (NVT_NH) [31], MD with multi-heat bath (NVT_VS2 or NVT_VS4), the geometry optimization by the steepest decent (SD) method (Opt), DIIS optimization method (DIIS), the eigenvector following (EF) method (EF) [63], and the rational function (RF) method (RF) [64] are available. For the details, see the Sections 'Geometry optimization' and 'Molecular dynamics'.

**MD.Fixed.XYZ**
In the geometry optimization and the molecular dynamics simulations, it is possible to separately fix the $x$-, $y$-, and $z$-coordinates of the atomic position to the initial position in your input file by the following keyword:

```
<MD.Fixed.XYZ
  1  1 1 1
  2  1 0 0
MD.Fixed.XYZ>
```

The example is for a system consisting of two atoms. If you have $N$ atoms, then you have to provide $N$ rows in this specification. The 1st column is the same sequential number to specify atom as in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. The 2nd, 3rd, and 4th columns are flags for the $x$-, $y$-, and $z$-coordinates, respectively. '1' means that the coordinate is fixed, and '0' relaxed. In the above example, the $x$-, $y$-, and $z$-coordinates of the atom '1' are fixed, only the $x$-coordinate of the atom '2' is fixed. The default setting is that all the coordinates are relaxed. The fixing of atomic positions are valid all the geometry optimizers and molecular dynamics schemes.

**MD.maxIter**
The keyword 'MD.maxIter' gives the number of MD iterations.

**MD.TimeStep**
The keyword 'MD.TimeStep' gives the time step (fs).

**MD.Opt.criterion**
When any of the geometry optimizers is chosen for the keyword 'MD.Type', then the keyword 'MD.Opt.criterion' specifies a convergence criterion (Hartree/Bohr). The geometry optimization is finished when a condition, the maximum force on atom is smaller than 'MD.Opt.criterion', is satisfied.

**MD.Opt.DIIS.History**
The keyword 'MD.Opt.DIIS.History' gives the number of previous steps to estimate the optimized structure used in the geometry optimization by 'DIIS', 'EF', and 'RF'. The default value is 3.

**MD.Opt.StartDIIS**
The geometry optimization step at which 'DIIS', 'EF', or 'RF' starts is specified by the keyword 'MD.Opt.StartDIIS'. The geometry optimization steps before starting the DIIS-type method is performed by the steepest decent method. The default value is 5.

**MD.TempControl**

The keyword specifies temperature for atomic motion in MD of the NVT ensembles. In 'NVT_VS', the temperature for nuclear motion can be controlled by

```
<MD.TempControl
  3
  100   2  1000.0  0.0
  400  10   700.0  0.4
  700  40   500.0  0.7
MD.TempControl>
```

The beginning of the description must be '<MD.TempControl', and the last of the description must be 'MD.TempControl>'. The first number '3' gives the number of the following lines to control the temperature. In this case, you can see that there are three lines. Following the number '3', in the consecutive lines the first column means MD steps and the second column gives the interval of MD steps that the velocity scaling is made. For the above example, a velocity scaling is performed at every two MD steps until 100 MD steps, at every 10 MD steps from 100 to 400 MD steps, and at every 40 MD steps from 400 to 700 MD steps. The third and fourth columns give a given temperature (K) and a scaling parameter $\alpha$ in the interval. For further details see the Section 'Molecular dynamics'. On the other hand, in NVT_NH, the temperature for nuclear motion can be controlled by

```
<MD.TempControl
  4
  1    1000.0
  100  1000.0
  400   700.0
  700   600.0
MD.TempControl>
```

The beginning of the description must be '<MD.TempControl', and the last of the description must be 'MD.TempControl>'. The first number '4' gives the number of the following lines to control the temperature. In this case you can see that there are four lines. Following the number '4', in the consecutive lines the first and second columns give the MD steps and a given temperature for nuclear motion. The temperature between the MD steps explicitly specified by the keyword is given by a linear interpolation.

**NH.Mass.HeatBath**

In 'NVT_NH', a mass of heat bath is given by the keyword. The default mass is 20, and the dimension is length$^2$ $\times$ mass. In this specification we use the bohr radius for the length, and the unified atomic mass unit, that the principal isotope of carbon atom is 12.0, for the mass.

**MD.Init.Velocity**

For molecular dynamics simulations, it is possible to provide the initial velocity of each atom by the following keyword:

```
<MD.Init.Velocity
```

```
 1    3000.000  0.0  0.0
 2   -3000.000  0.0  0.0
MD.Init.Velocity>
```

The example is for a system consisting of two atoms. If you have $N$ atoms, then you have to provide $N$ rows in this specification. The 1st column is the same sequential number to specify atom as in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. The 2nd, 3rd, and 4th columns are $x$-, $y$-, and $z$-components of the velocity of each atom. The unit of the velocity is m/s. The keyword 'MD.Init.Velocity' is compatible with the keyword 'MD.Fixed.XYZ'.

## Band dispersion

**Band.dispersion**
When you evaluate the band dispersion, please specify 'ON' for the keyword 'Band.dispersion'.

**Band.KPath.UnitCell**
The keyword 'Band.KPath.UnitCell' gives unit vectors, which are used in the calculation of the band dispersion, as follows:

```
<Band.KPath.UnitCell
 3.56 0.0 0.0
 0.0 3.56 0.0
 0.0 0.0 3.56
Band.KPath.UnitCell>
```

The beginning of the description must be '<Band.KPath.UnitCell', and the last of the description must be 'Band.KPath.UnitCell>'. If 'Band.KPath.UnitCell' exists, the reciprocal lattice vectors for the calculation of the band dispersion are calculated by the unit vectors specified in 'Band.KPath.UnitCell'. If 'Band.KPath.UnitCell' is not found, the reciprocal lattice vectors, which are calculated by the unit vectors specified in 'Atoms.UnitVectors', is employed for the calculation of the band dispersion. In case of fcc, bcc, base centered cubic, and trigonal cells, the reciprocal lattice vectors for the calculation of the band dispersion should be specified using the keyword 'Band.KPath.UnitCell' based on the consuetude in the band calculations.

**Band.Nkpath**
The keyword 'Band.Nkpath' gives the number of paths for the band dispersion.

**Band.kpath**
The keyword 'Band.kpath' specifies the paths of the band dispersion as follows:

```
<Band.kpath
  15  0.0 0.0 0.0   1.0 0.0 0.0   g X
  15  1.0 0.0 0.0   1.0 0.5 0.0   X W
  15  1.0 0.5 0.0   0.5 0.5 0.5   W L
  15  0.5 0.5 0.5   0.0 0.0 0.0   L g
  15  0.0 0.0 0.0   1.0 1.0 0.0   g X
Band.kpath>
```

The beginning of the description must be '<Band.kpath', and the last of the description must be 'Band.kpath>'. The number of lines should be consistent with 'Band.Nkpath'. The first column is the number of grids at which eigenvalues are evaluated on the path. The following (n1, n2, n3) and (n1', n2', n3'), spanned by the reciprocal lattice vectors, specifies the starting and ending **k**-points of the path in the first Brillouin zone. If 'Band.KPath.UnitCell' is found, the reciprocal lattice vectors for the calculation of the band dispersion are calculated by the unit vectors specified in 'Band.KPath.UnitCell'. If 'Band.KPath.UnitCell' is not found, the reciprocal lattice vectors, which

are calculated by the unit vectors specified in 'Atoms.UnitVectors' is employed for the calculation of the band dispersion. The final two alphabets give the name of the starting and ending **k**-points of the path.

## Restarting

**scf.restart**
If you want to restart the SCF calculation using a previous file '*System.Name*_rst/*' which should be generated in the previous calculation, then set the keyword 'scf.restart' to 'ON'.

## Output of molecular orbitals (MOs)

**MO.fileout**
If you want to output molecular orbitals (MOs) to files, then set the keyword 'MO.fileout' to 'ON'.

**num.HOMOs**
The keyword 'num.HOMOs' gives the number of the highest occupied molecular orbitals (HOMOs) that you want to output to files.

**num.LUMOs**
The keyword 'num.LUMOs' gives the number of the lowest unoccupied molecular orbitals (LUMOs) that you want to output to files.

**MO.Nkpoint**
When you have specified 'MO.fileout=ON' and 'scf.EigenvalueSolver=Band', the keyword 'MO.Nkpoint' gives the number of the **k**-points at which you output MOs to files.

**MO.kpoint**
The keyword 'MO.kpoint' specifies the **k**-point, at which MOs are evaluated for the output to files, as follows:

```
<MO.kpoint
  0.0  0.0  0.0
MO.kpoint>
```

The beginning of the description must be '<MO.kpoint', and the last of the description must be 'MO.kpoint>'. The k-points are specified by (n1, n2, n3) which is spanned by the reciprocal lattice vectors, where the the reciprocal lattice vectors are determined in the same way as 'Band.kpath'.

## DOS and PDOS

**Dos.fileout**
If you want to evaluate density of states (DOS) and projected partial density of states (PDOS), please set in 'Dos.fileout=ON'.

**Dos.Erange**
The keyword 'Dos.Erange' determines the energy range for the DOS calculation as

```
Dos.Erange                -10.0  10.0
```

The first and second values are the lower and upper bounds of the energy range (eV) for the DOS calculation, respectively.

**Dos.Kgrid**
The keyword 'Dos.Kgrid' gives a set of numbers (n1,n2,n3) of grids to discretize the first Brillouin zone in the **k**-space, which is used in the DOS calculation.

## Interface for developers

**HS.fileout**
If you want to use Kohn-Sham Hamiltonian, overlap, and density matrices, please set in 'HS.fileout=ON'. Then, these data are stored to '*System.Name*.scfout' in a binary form, where '*System.Name*' is the file name specified by the keyword 'System.Name'. The utilization of these data is illustrated in the Section 'Interface for developers'.

## Voronoi charge

**Voronoi.charge**
If you want to calculate Voronoi charges, then set the keyword 'Voronoi.charge' to 'ON'. The result is found in '*System.Name*.out', '*System.Name*' is the file name specified by the keyword 'System.Name'.

# 8 Output files

In case of 'level.of.fileout=0', the following files are generated. In the following, '*System.Name*' is the file name specified by the keyword 'System.Name'.

- *System.Name*.out

  The history of SCF calculations, the history of geometry optimization, Mulliken charges, the total energy, and the dipole moment.

- *System.Name*.xyz

  The final geometrical structure obtained by MD or the geometry optimization, which can be read by OpenMX Viewer [152, 151] and XCrySDen [105].

- *System.Name*.bulk.xyz

  If 'scf.EigenvalueSolver=Band', atomic coordinates including atoms in copied cells are output, which can be read by OpenMX Viewer [152, 151] and XCrySDen [105].

- *System.Name*_rst/

  The directory storing restart files.

- *System.Name*.md

  Geometrical coordinates at every MD step in the xyz format, which can be read by OpenMX Viewer [152, 151].

- *System.Name*.md2

  Geometrical coordinates at the final MD step with the species names that you specified .

- *System.Name*.cif

  Initial geometrical coordinates in the cif format.

- *System.Name*.ene

  Values computed at every MD step. The values are found in the routine 'iterout.c'.

In case of 'level.of.fileout=1', the following Gaussian cube files are generated, in addition to files generated in 'level.of.fileout=0', In the following, '*System.Name*' is the file name specified by the keyword 'System.Name'.

- *System.Name*.tden.cube

  Total electron density in the Gaussian cube format.

- *System.Name*.sden.cube

  If the spin-polarized calculation using 'LSDA-CA', 'LSDA-PW', or 'GGA-PBE' is performed, then spin electron density is output in the Gaussian cube format.

- *System.Name*.dden.cube

  Difference electron density taken from superposition of atomic densities of constituent atoms in the Gaussian cube format.

- *System.Name*.v0.cube

  The Kohn-Sham potential excluding the non-local potential for up-spin in the Gaussian cube format. If the projector expansion method is switched on by the keyword 'scf.ProExpn.VNA', the VNA potential is also excluded. See also the technical note 'Total Energy and Forces' at http://www.openmx-square.org/tech_notes/tech_notes.html .

- *System.Name*.v1.cube

  The Kohn-Sham potential excluding the non-local potential for down-spin in the Gaussian cube format in the spin-polarized calculation. If the projector expansion method is switched on by the keyword 'scf.ProExpn.VNA', the VNA potential is also excluded. See also the technical note 'Total Energy and Forces' at http://www.openmx-square.org/tech_notes/tech_notes.html .

- *System.Name*.vhart.cube

  The Hartree potential calculated by the difference charge density in the Gaussian cube format. See also the technical note 'Total Energy and Forces' at http://www.openmx-square.org/tech_notes/tech_notes.html .

In case of 'level.of.fileout=2', the following files are generated in addition to files generated in level.of.fileout=1, In the following, '*System.Name*' is the file name specified by the keyword 'System.Name'.

- *System.Name*.vxc0.cube

  The exchange-correlation potential for up-spin in the Gaussian cube format.

- *System.Name*.vxc1.cube

  The exchange-correlation potential for down-spin in the Gaussian cube format.

- *System.Name*.grid

  The real space grids which are used numerical integrations and the solution of Poisson's equation.

If 'MO.fileout=ON' and 'scf.EigenvalueSolver=Cluster', the following files are also generated:

- *System.Name*.homo0_0.cube, *System.Name*.homo0_1.cube, ...

  The HOMOs are output in the Gaussian cube format. The first number below 'homo' means a spin state (up=0, down=1). The second number specifies the eigenstates, i.e., 0, 1, and 2 correspond to HOMO, HOMO-1, and HOMO-2, respectively.

- *System.Name*.lumo0_0.cube, *System.Name*.lumo0_1.cube, ...

  The LUMOs are output in the Gaussian cube format. The first number below 'lumo' means a spin state (up=0, down=1). The second number specifies the eigenstates, i.e., 0, 1, and 2 correspond to LUMO, LUMO+1, and LUMO+2, respectively.

If 'MO.fileout=ON' and 'scf.EigenvalueSolver=Band', the following files are also generated:

- *System.Name*.homo0_0_0_r.cube, *System.Name*.homo1_0_1_r.cube, ... *System.Name*.homo0_0_0_i.cube, *System.Name*.homo1_0_1_i.cube, ...

  The HOMOs are output in the Gaussian cube format. The first number below 'homo' means the k-point number, which is specified by the keyword 'MO.kpoint'. The second number is a spin state (up=0, down=1). The third number specifies the eigenstates, i.e., 0, 1, and 2 correspond to HOMO, HOMO-1, and HOMO-2, respectively. The 'r' and 'i' mean the real and imaginary parts of the wave function.

- *System.Name*.lumo0_0_0_r.cube, *System.Name*.lumo1_0_1_r.cube, ... *System.Name*.lumo0_0_0_i.cube, *System.Name*.lumo1_0_1_i.cube, ...

  The LUMOs are output in the Gaussian cube format. The first number below 'lumo' means the k-point number, which is specified in the keyword, MO.kpoint. The second number is a spin state (up=0, down=1). The third number specifies the eigenstates, i.e., 0, 1, and 2 correspond to LUMO, LUMO+1, and LUMO+2, respectively. The 'r' and 'i' mean the real and imaginary parts of the wave function.

If 'Band.Nkpath' is not 0 and 'scf.EigenvalueSolver=Band', the following file is also generated:

- *System.Name*.Band

  A data file for the band dispersion.

If 'Dos.fileout=ON', the following files are also generated:

- *System.Name*.Dos.val

  A data file of eigenvalues for calculating the density of states.

- *System.Name*.Dos.vec

  A data file of eigenvectors for calculating the density of states.

If 'scf.SpinPolarization=NC' and 'level.of.fileout=1' or '2', the following files are also generated:

- *System.Name*.nco.xsf

  A vector file which stores a non-collinear orbital moment projected on each atom by means of Mulliken analysis, which can be visualized using 'Display→Forces' in XCrySDen.

- *System.Name*.nc.xsf

  A vector file which stores a non-collinear spin moment projected on each atom by means of Mulliken analysis, which can be visualized using 'Display→Forces' in XCrySDen.

- *System.Name*.ncsden.xsf

  A vector file which stores a non-collinear spin moment on real space grids, which can be visualized using 'Display→Forces' in XCrySDen.

# 9 Functional

In OpenMX, local density approximations (LDA, LSDA) [2, 3, 4] and a generalized gradient approximation (GGA) [5] to exchange-correlation functional are used. Using a keyword 'scf.XcType', you can choose one of approximations to the exchange-correlation functional:

```
scf.XcType                  LDA        # LDA|LSDA-CA|LSDA-PW|GGA-PBE
```

Currently, 'LDA', 'LSDA-CA', 'LSDA-PW', and 'GGA-PBE' are available, where 'LSDA-CA' is the local spin density functional of Ceperley-Alder [2], 'LSDA-PW' is the local spin density functional of Perdew-Wang, in which the gradient of density is set in zero in their GGA formalism [4]. Note: 'LSDA-CA' is faster than 'LSDA-PW'. 'GGA-PBE' is GGA proposed by Perdew, Burke, and Ernzerhof [5]. The GGA is implemented by using the first order finite difference method in real space. In addition, LDA+$U$ (or GGA+$U$) functionals are also available. For the details, see the Section 'DFT+$U$'. The relevant keyword to specify the spin (un)polarized and non-collinear calculations is 'scf.SpinPolarization'.

```
scf.SpinPolarization        off        # On|Off|NC
```

If the calculation for the spin polarization is performed, then specify 'ON'. If the calculation for the non-spin polarization is performed, then specify 'OFF'. When you use 'LDA' for the keyword 'scf.XcType', the keyword 'scf.SpinPolarization' must be off. In addition to these options, 'NC' is supported for the non-collinear DFT calculation. For this calculation, see also the Section 'Non-collinear DFT'.

# 10 Basis sets

## 10.1 General

OpenMX uses numerical pseudo-atomic orbitals (PAOs) $\chi$ as basis function to expand one-particle Kohn-Sham wave functions. The PAO function is given by a product of a radial function $R$ and a real spherical harmonic function $Y$ as

$$\chi(\mathbf{r}) = R(r)Y(\hat{\mathbf{r}}),$$

where the radial function $R$ is a numerically defined one, and finite within a cutoff radius in real space. In other words, the function $R$ becomes zero beyond a pre-defined cutoff radius. The PAO function calculated by ADPACK is called *primitive* function, and an *optimized* PAO function is obtained by the orbital optimization method in OpenMX starting from the primitive PAO function [41]. They are stored in a file with a file extension of 'pao'. When the OpenMX calculation is performed, the numerical data stored in the file are read, and the value at any $r$ is obtained by an interpolation technique. The files with the file extension of 'pao' should be stored in a directory, e.g., 'DFT_DATA19/PAO', where the directory without 'PAO' can be specified by the following keyword:

```
DATA.PATH    ../DFT_DATA19    # default=../DFT_DATA19
```

Both the absolute and relative specifications are possible, and the default is '../DFT_DATA19'.

In an input file for the OpenMX calculation, The basis set is specified by a keyword 'Definition.of.Atomic.Species' as follows:

```
<Definition.of.Atomic.Species
  H   H5.0-s2p1      H_PBE19
  C   C5.0-s2p1      C_PBE19
Definition.of.Atomic.Species>
```

where an abbreviation, H5.0-s2p1, of the basis function is introduced. H5.0 stands for the file name of the PAO functions without the file extension which must exist in a directory specified by the keyword 'DATA.PATH', e.g., DFT_DATA19/PAO, and 5.0 implies the cutoff radius of the PAO functions. Also, s2p1 means that two s-state radial functions and one p-state radial function stored in the file are used. In this case, totally five PAO basis functions (2x1+1x3=5) are assigned for 'H'.

Since optimized basis functions are available on the website (http://www.openmx-square.org/) as the database Ver. 2019. We recommend for general users to use these optimized basis functions. But for experts, both the primitive and optimized PAO functions are explained in the subsequent sections.

## 10.2 Primitive basis functions

The primitive basis functions are generated by ADPACK, and they are the ground and exited states of a pseudo-atom with a confinement pseudopotential [41] as shown in Fig. 1. The functions are numerical table function stored in a file of which file extension is 'pao'. You will see that the ground state is nodeless and the first exited state has one node, and the number of nodes increases in the further excited states. When you use the primitive PAO functions as basis set, the one-particle Kohn-Sham functions are expressed by the linear combination of the pseudo-atomic type basis functions where

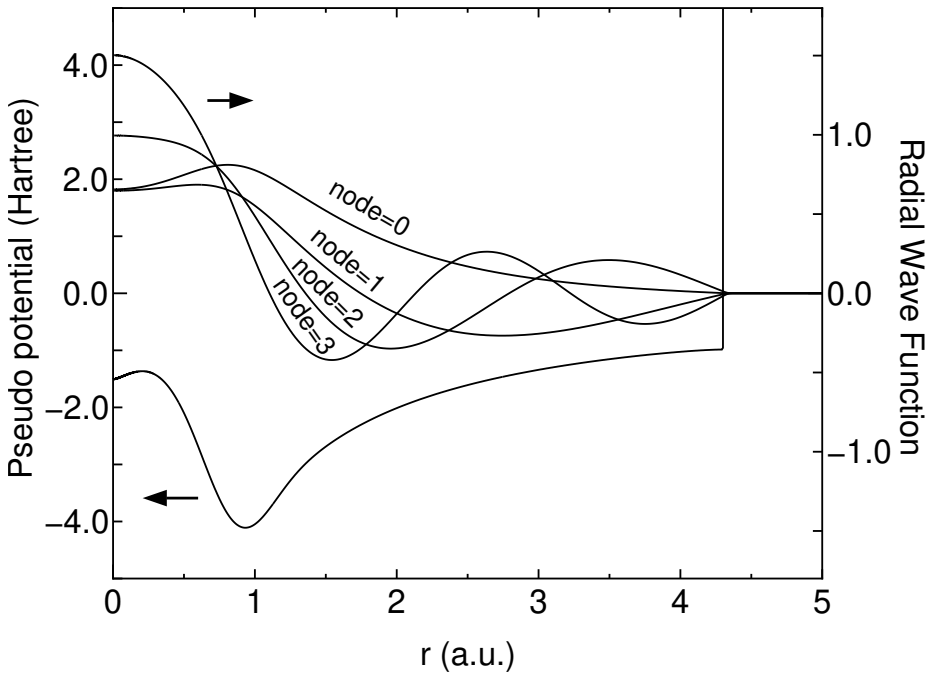



Figure 1: Primitive basis functions for s-orbitals of a carbon pseudo-atom with a confinement pseudopotential.

each basis function is the product of the radial function and a real spherical harmonics function. The accuracy and efficiency of the calculations can be controlled by two parameters: a cutoff radius and the number of basis functions. In general, one can get the convergent results by increasing the cutoff radius and the number of basis functions as shown in Fig. 2. However, it is noted that the use of a large number of basis orbitals with a large cutoff radius requires an extensive computational resource such as memory size and computational time. The general trend to choose the cutoff radius and the number of basis orbitals in a compromising way is discussed in Ref. [41], where you may find that basis orbitals with a higher angular momentum are needed to achieve the sufficient convergence for elements, such as F and Cl, in the right hand side of the periodic table, and that a large cutoff radius of basis orbitals should be used for elements, such as Li and Na, in the left hand side of the periodic table. Since optimized basis functions are available on the website (http://www.openmx-square.org/) as the database Ver. 2019. We recommend for general users to use these optimized basis functions instead of the primitive PAO functions.

### 10.3 Optimized basis functions provided by the database Ver. 2019

The optimized PAO functions are provided on the website (http://www.openmx-square.org/) as the database Ver. 2019. This should be the first choice by general users, since they were generated by the orbital optimization method [41], and tested well through a series of benchmark calculations. For most elements in the database Ver. 2019, three systems are chosen as training sets of chemical environment, and the PAO functions were optimized by the orbital optimization method for the chosen systems [41]. Then, those optimized ones are unified to form a single PAO file through a combination scheme of a subspace rotation method and Gram-Schmidt orthogonalization. Thus, the optimized PAO functions

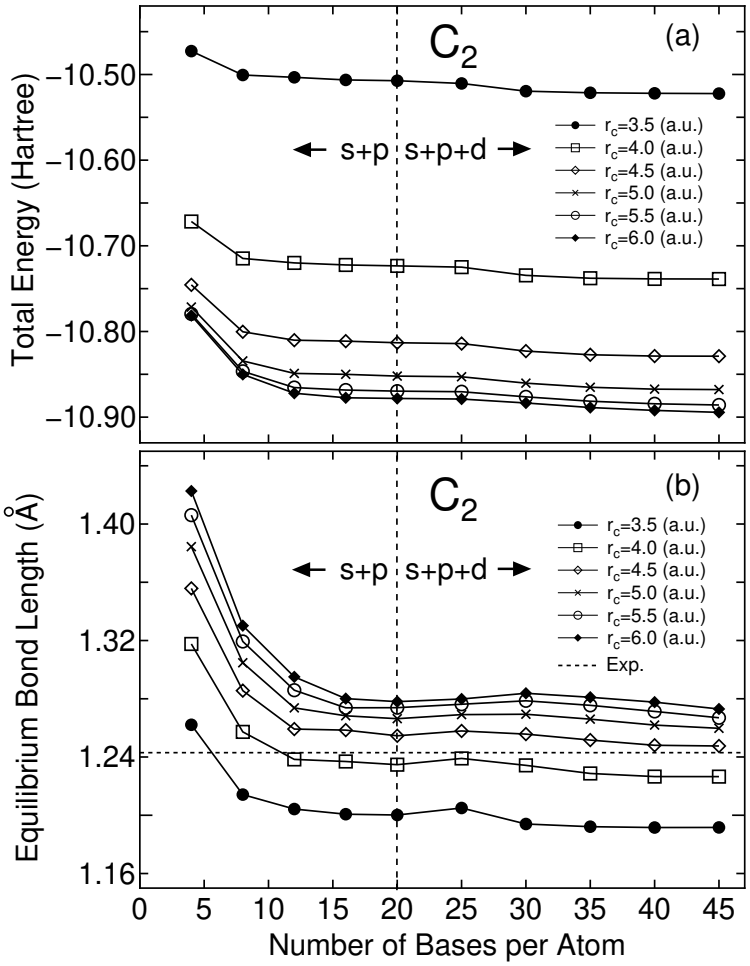



Figure 2: Convergence properties of (a) the total energy and (b) the equilibrium bond length for a carbon dimer with respect to the cutoff radius and the number of basis functions.

have been already optimized for a set of different chemical environments, which may increase the transferability of the optimized PAO functions. In fact, the series of benchmark calculations shown in the website of the database are in good agreement with corresponding all electron calculations. From the benchmark calculations one may find a proper cutoff radius and the number of basis functions for each element. The input files used for the benchmark calculations are also available on the website, which may be useful for users to get used to the OpenMX calculations at the initial stage. The accuracy of the database (2019) was validated by the delta gauge [40]. The mean delta factor of 71 elements is 1.774 meV/atom with the standard deviation of 1.702 meV/atom, which implies high accuracy of the database (2019). Users are strongly encouraged to use the new database due to the high accuracy. See also the section 'Calculation of Energy vs. lattice constant'.

For user's convenience, a set of proper choices for the basis functions is provided in Tables 1 and 2. For each pseudopotential, three choices: *Quick*, *Standard*, and *Precise* are given in the Tables. *Quick* allows us a quick calculation, but with a reasonably accuracy. *Standard* can be a proper choice for the most of users, which balances the accuracy and efficiency. *Precise* almost reaches to convergent results in most of cases. The tables give a guideline in choosing the basis functions, while of course,

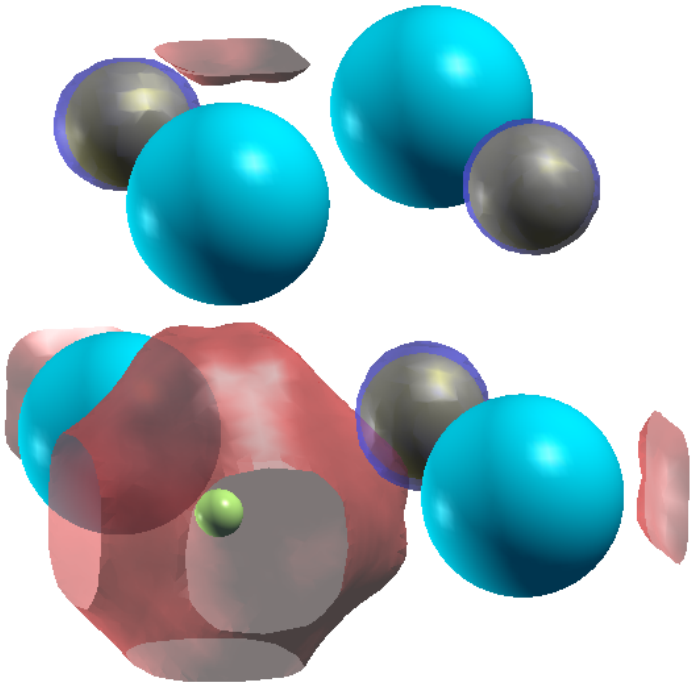


Figure 3: The isosurface map of the highest occupied state at the Γ point for NaCl with a Cl-site vacancy, which shows a F-center in NaCl with a Cl vacancy. The isosurface map was drawn using XCrySDen with the isovalue of 0.042 [105]. The calculation was done with the system charge of -1 using a keyword 'scf.system.charge'. The watery and silver colors correspond to sodium and chlorine atoms,respectively, and the yellow small ball shows the position of empty atom.

basis functions should be properly selected depending on your purpose.

### 10.4 Optimization of PAO by yourself

Starting from the primitive basis functions, you can optimize the radial shape variationally so that the accuracy can be increased. See the details in the Section 'Orbital optimization'.

### 10.5 Empty atom scheme

The primitive and optimized PAO functions are usually assigned to atoms. Moreover, it is possible to assign basis functions in any vacant region using an *empty* atom. You will find the empty atom 'E' in the website of the database (http://www.openmx-square.org/). Using the pseudopotential for the empty atom 'E', though the pseudopotential is a flat zero potential, you can put basis functions at any place independently of atomic position. To do that, you can define empty atoms by

```
<Definition.of.Atomic.Species
  H     H5.0-s2p1       H_PBE19
  C     C5.0-s2p1       C_PBE19
  EH    H5.0-s2p1       E
```

```
  EC   C5.0-s2p1      E
Definition.of.Atomic.Species>
```

In the example, two sorts of empty atoms are defined as 'EH' and 'EC' which have basis sets specified by 'H5.0-s2p1' and 'C5.0-s2p1', respectively, which means that one can use any basis functions for an empty atom as shown above. Then 'EH' and 'EC' can be put to any place by the keyword 'Atoms.SpeciesAndCoordinates', where the number of electrons for the empty atom is zero. To define an empty atom, only thing you have to do is to use 'E.vps' as pseudopotential for the empty atom. The empty atom scheme enables us not only to estimate the basis set superposition error (BSSE) using the counterpoise correction (CP) method [46, 47], but also to treat a vacancy state and a nearly free electron state on metal surfaces within the linear combination of pseudo-atomic orbitals (LCPAO) method. As an example, a calculation of a F-center in NaCl with a Cl vacancy is shown in Fig. 3. We see that the highest occupied state at the $\Gamma$ point is the F-center state. You can follow the calculation using 'NaCl_FC.dat' in the directory 'work'.

### 10.6 Specification of a directory storing PAO and VPS files

The path to the VPS and PAO directories can be specified in your input file by the following keyword:

```
DATA.PATH    ../DFT_DATA19    # default=../DFT_DATA19
```

Both the absolute and relative specifications are possible. PAO files in a database should not be used for the VPS in other databases, since semicore states included in several elements are different from each other. So, the consistency in the version of PAO and VPS must be kept. For that reason, it would be better to store PAO and VPS files of each version in different directories. In this case, the keyword is useful.

Table 1: A set of choices for the PAO basis functions from E to Kr.

| VPS | Valence electrons | Quick | Standard | Precise |
|---|---|---|---|---|
| E | 0.0 | Kr10.0-s1p1 | Kr10.0-s2p1d1 | Kr10.0-s2p2d1f1 |
| H_PBE19 | 1.0 | H5.0-s2 | H6.0-s2p1 | H7.0-s2p2d1 |
| He_PBE19 | 2.0 | He8.0-s1p1 | He8.0-s2p1 | He10.0-s2p2d1 |
| Li_PBE19 | 3.0 | Li8.0-s3p1 | Li8.0-s3p2 | Li8.0-s3p2d1 |
| Be_PBE19 | 2.0 | Be7.0-s2p1 | Be7.0-s2p2 | Be7.0-s3p2d1 |
| B_PBE19 | 3.0 | B7.0-s2p2 | B7.0-s2p2d1 | B7.0-s3p2d2 |
| C_PBE19 | 4.0 | C6.0-s2p2 | C6.0-s2p2d1 | C6.0-s3p2d2 |
| N_PBE19 | 5.0 | N6.0-s2p2 | N6.0-s2p2d1 | N6.0-s3p2d2 |
| O_PBE19 | 6.0 | O6.0-s2p2 | O6.0-s2p2d1 | O6.0-s3p2d2 |
| F_PBE19 | 7.0 | F6.0-s2p2 | F6.0-s2p2d1 | F6.0-s3p3d2f1 |
| Ne_PBE19 | 8.0 | Ne9.0-s2p2 | Ne9.0-s2p2d1 | Ne9.0-s3p2d2 |
| Na_PBE19 | 9.0 | Na9.0-s3p2 | Na9.0-s3p2d1 | Na9.0-s3p2d2 |
| Mg_PBE19 | 8.0 | Mg9.0-s2p2 | Mg9.0-s3p2d1 | Mg9.0-s3p2d2 |
| Al_PBE19 | 3.0 | Al7.0-s2p1d1 | Al7.0-s2p2d1 | Al7.0-s3p2d2 |
| Si_PBE19 | 4.0 | Si7.0-s2p1d1 | Si7.0-s2p2d1 | Si7.0-s3p3d2 |
| P_PBE19 | 5.0 | P7.0-s2p2d1 | P7.0-s2p2d1f1 | P7.0-s3p2d2f1 |
| S_PBE19 | 6.0 | S7.0-s2p2d1 | S7.0-s2p2d1f1 | S7.0-s3p2d2f1 |
| Cl_PBE19 | 7.0 | Cl7.0-s2p2d1 | Cl7.0-s2p2d1f1 | Cl7.0-s3p2d2f1 |
| Ar_PBE19 | 8.0 | Ar9.0-s2p2d1 | Ar9.0-s2p2d1f1 | Ar9.0-s3p2d2f1 |
| K_PBE19 | 9.0 | K10.0-s3p2 | K10.0-s3p2d1 | K10.0-s3p2d2 |
| Ca_PBE19 | 10.0 | Ca9.0-s3p2 | Ca9.0-s3p2d1 | Ca9.0-s3p2d2 |
| Sc_PBE19 | 11.0 | Sc9.0-s2p2d1 | Sc9.0-s3p2d1 | Sc9.0-s3p2d2 |
| Ti_PBE19 | 12.0 | Ti7.0-s2p2d1 | Ti7.0-s3p2d1 | Ti7.0-s3p2d2f1 |
| V_PBE19 | 13.0 | V6.0-s2p2d1 | V6.0-s3p2d1 | V6.0-s3p2d2f1 |
| Cr_PBE19 | 14.0 | Cr6.0-s2p2d1 | Cr6.0-s3p2d1 | Cr6.0-s3p2d2f1 |
| Mn_PBE19 | 15.0 | Mn6.0-s2p2d1 | Mn6.0-s3p2d1 | Mn6.0-s3p2d2f1 |
| Fe_PBE19H | 16.0 | Fe5.5H-s2p2d1 | Fe5.5H-s3p2d1 | Fe5.5H-s3p2d2f1 |
| Fe_PBE19S | 14.0 | Fe6.0S-s2p2d1 | Fe6.0S-s3p2d1 | Fe6.0S-s3p2d2f1 |
| Co_PBE19H | 17.0 | Co6.0H-s2p2d1 | Co6.0H-s3p2d1 | Co6.0H-s3p2d2f1 |
| Co_PBE19S | 15.0 | Co6.0S-s2p2d1 | Co6.0S-s3p2d1 | Co6.0S-s3p2d2f1 |
| Ni_PBE19H | 18.0 | Ni6.0H-s2p2d1 | Ni6.0H-s3p2d1 | Ni6.0H-s3p2d2f1 |
| Ni_PBE19S | 16.0 | Ni6.0S-s2p2d1 | Ni6.0S-s3p2d1 | Ni6.0S-s3p2d2f1 |
| Cu_PBE19H | 19.0 | Cu6.0H-s2p2d1 | Cu6.0H-s3p2d1 | Cu6.0H-s3p2d2f1 |
| Cu_PBE19S | 11.0 | Cu6.0S-s2p1d1 | Cu6.0S-s3p2d1 | Cu6.0S-s3p2d2f1 |
| Zn_PBE19H | 20.0 | Zn6.0H-s2p2d1 | Zn6.0H-s3p2d1 | Zn6.0H-s3p2d2f1 |
| Zn_PBE19S | 12.0 | Zn6.0S-s2p1d1 | Zn6.0S-s3p2d1 | Zn6.0S-s3p2d2f1 |
| Ga_PBE19 | 13.0 | Ga7.0-s2p2d1 | Ga7.0-s3p2d2 | Ga7.0-s3p2d2f1 |
| Ge_PBE19 | 4.0 | Ge7.0-s2p1d1 | Ge7.0-s3p2d2 | Ge7.0-s3p2d2f1 |
| As_PBE19 | 15.0 | As7.0-s3p2d1 | As7.0-s3p2d2 | As7.0-s3p2d2f1 |
| Se_PBE19 | 6.0 | Se7.0-s3p2d1 | Se7.0-s3p2d2 | Se7.0-s3p2d2f1 |
| Br_PBE19 | 7.0 | Br7.0-s3p2d1 | Br7.0-s3p2d2 | Br7.0-s3p2d2f1 |
| Kr_PBE19 | 8.0 | Kr10.0-s2p2d1 | Kr10.0-s3p2d2 | Kr10.0-s3p2d2f1 |

Table 2: A set of choices for the PAO basis functions from Rb to Bi.

| VPS | Valence electrons | Quick | Standard | Precise |
|---|---|---|---|---|
| Rb_PBE19 | 9.0 | Rb11.0-s2p2d1 | Rb11.0-s3p2d2 | Rb11.0-s3p2d2f1 |
| Sr_PBE19 | 10.0 | Sr10.0-s2p2d1 | Sr10.0-s3p2d2 | Sr10.0-s3p3d2f1 |
| Y_PBE19 | 11.0 | Y10.0-s3p2d1 | Y10.0-s3p2d2 | Y10.0-s3p3d2f1 |
| Zr_PBE19 | 12.0 | Zr7.0-s3p2d1 | Zr7.0-s3p2d2 | Zr7.0-s3p2d2f1 |
| Nb_PBE19 | 13.0 | Nb7.0-s3p2d1 | Nb7.0-s3p2d2 | Nb7.0-s3p2d2f1 |
| Mo_PBE19 | 14.0 | Mo7.0-s3p2d1 | Mo7.0-s3p2d2 | Mo7.0-s3p2d2f1 |
| Tc_PBE19 | 15.0 | Tc7.0-s3p2d1 | Tc7.0-s3p2d2 | Tc7.0-s3p2d2f1 |
| Ru_PBE19 | 14.0 | Ru7.0-s3p2d1 | Ru7.0-s3p2d2 | Ru7.0-s3p2d2f1 |
| Rh_PBE19 | 15.0 | Rh7.0-s3p2d1 | Rh7.0-s3p2d2 | Rh7.0-s3p2d2f1 |
| Pd_PBE19 | 16.0 | Pd7.0-s3p2d1 | Pd7.0-s3p2d2 | Pd7.0-s3p2d2f1 |
| Ag_PBE19 | 17.0 | Ag7.0-s3p2d1 | Ag7.0-s3p2d2 | Ag7.0-s3p2d2f1 |
| Cd_PBE19 | 12.0 | Cd7.0-s3p2d1 | Cd7.0-s3p2d2 | Cd7.0-s3p2d2f1 |
| In_PBE19 | 13.0 | In7.0-s3p2d1 | In7.0-s3p2d2 | In7.0-s3p2d2f1 |
| Sn_PBE19 | 14.0 | Sn7.0-s3p2d1 | Sn7.0-s3p2d2 | Sn7.0-s3p2d2f1 |
| Sb_PBE19 | 15.0 | Sb7.0-s3p2d1 | Sb7.0-s3p2d2 | Sb7.0-s3p2d2f1 |
| Te_PBE19 | 16.0 | Te7.0-s3p2d2 | Te7.0-s3p2d2f1 | Te7.0-s3p3d2f1 |
| I_PBE19 | 7.0 | I7.0-s3p2d2 | I7.0-s3p2d2f1 | I7.0-s3p3d2f1 |
| Xe_PBE19 | 8.0 | Xe11.0-s3p2d1 | Xe11.0-s3p2d2 | Xe11.0-s3p2d2f1 |
| Cs_PBE19 | 9.0 | Cs12.0-s3p2d1 | Cs12.0-s3p2d2 | Cs12.0-s3p2d2f1 |
| Ba_PBE19 | 10.0 | Ba10.0-s3p2d1 | Ba10.0-s3p2d2 | Ba10.0-s3p2d2f1 |
| La_PBE19 | 11.0 | La8.0-s3p2d1f1 | La8.0-s3p2d2f1 | La8.0-s3p3d2f1 |
| Ce_PBE19 | 12.0 | Ce8.0-s3p2d1f1 | Ce8.0-s3p2d2f1 | Ce8.0-s3p3d2f1 |
| Pr_PBE19 | 13.0 | Pr8.0-s3p2d1f1 | Pr8.0-s3p2d2f1 | Pr8.0-s3p3d2f1 |
| Nd_PBE19 | 14.0 | Nd8.0-s3p2d1f1 | Nd8.0-s3p2d2f1 | Nd8.0-s3p3d2f1 |
| Pm_PBE19 | 15.0 | Pm8.0-s3p2d1f1 | Pm8.0-s3p2d2f1 | Pm8.0-s3p3d2f1 |
| Sm_PBE19 | 16.0 | Sm8.0-s3p2d1f1 | Sm8.0-s3p2d2f1 | Sm8.0-s3p3d2f1 |
| Dy_PBE19 | 20.0 | Dy8.0-s3p2d1f1 | Dy8.0-s3p2d2f1 | Dy8.0-s3p3d2f1 |
| Ho_PBE19 | 21.0 | Ho8.0-s3p2d1f1 | Ho8.0-s3p2d2f1 | Ho8.0-s3p3d2f1 |
| Lu_PBE19 | 11.0 | Lu8.0-s3p2d2 | Lu8.0-s3p2d2f1 | Lu8.0-s3p3d2f1 |
| Hf_PBE19 | 12.0 | Hf9.0-s3p2d2 | Hf9.0-s3p2d2f1 | Hf9.0-s3p3d2f1 |
| Ta_PBE19 | 13.0 | Ta7.0-s3p2d2 | Ta7.0-s3p2d2f1 | Ta7.0-s3p3d2f1 |
| W_PBE19 | 12.0 | W7.0-s3p2d2 | W7.0-s3p2d2f1 | W7.0-s3p3d2f1 |
| Re_PBE19 | 15.0 | Re7.0-s3p2d2 | Re7.0-s3p2d2f1 | Re7.0-s3p3d2f1 |
| Os_PBE19 | 14.0 | Os7.0-s3p2d2 | Os7.0-s3p2d2f1 | Os7.0-s3p3d2f1 |
| Ir_PBE19 | 15.0 | Ir7.0-s3p2d2 | Ir7.0-s3p2d2f1 | Ir7.0-s3p3d2f1 |
| Pt_PBE19 | 16.0 | Pt7.0-s3p2d2 | Pt7.0-s3p2d2f1 | Pt7.0-s3p3d2f1 |
| Au_PBE19 | 17.0 | Au7.0-s3p2d2 | Au7.0-s3p2d2f1 | Au7.0-s3p3d2f1 |
| Hg_PBE19 | 18.0 | Hg8.0-s3p2d2 | Hg8.0-s3p2d2f1 | Hg8.0-s3p3d2f1 |
| Tl_PBE19 | 19.0 | Tl8.0-s3p2d2 | Tl8.0-s3p2d2f1 | Tl8.0-s3p3d2f1 |
| Pb_PBE19 | 14.0 | Pb8.0-s3p2d2 | Pb8.0-s3p2d2f1 | Pb8.0-s3p3d2f1 |
| Bi_PBE19 | 15.0 | Bi8.0-s3p2d2 | Bi8.0-s3p2d2f1 | Bi8.0-s3p3d2f1 |

# 11 Pseudopotentials

## 11.1 Conventional pseudopotentials

The core Coulomb potential in OpenMX is replaced by a tractable norm-conserving pseudopotential proposed by Morrison, Bylander, and Kleinman [36], which is a norm-conserving version of the ultrasoft pseudopotential by Vanderbilt [37]. Although the pseudopotentials can be generated using ADPACK which is a program package for atomic density functional calculations and available from a website (http://www.openmx-square.org/), for your convenience, we offer a database (http://www.openmx-square.org/) of the pseudopotentials as the database Ver. 2019. If you want to use pseudopotentials stored in the database, then copy them to the directory, 'openmx3.9/DFT_DATA19/VPS/', while most of data have been already copied in the distributed package of OpenMX Ver. 3.9. You can freely utilize these data, but we cannot offer any warranty on these data. The assignation of pseudopotentials can be made using a keyword 'Definition.of.Atomic.Species' as in the case of specification of basis functions as follows:

```
<Definition.of.Atomic.Species
  H   H6.0-s2p1        H_CA19
  C   C6.0-s2p2        C_CA19
Definition.of.Atomic.Species>
```

The pseudopotential file can be specified in the third column, and the file must be existing in the directory 'DFT_DATA19/VPS'. In the specification of atomic coordinates, it is required to give the number of electrons for up- and down-spin states for each atom as follows:

```
<Atoms.SpeciesAndCoordinates
  1   C      0.000000    0.000000    0.000000     2.0  2.0
  2   H     -0.889981   -0.629312    0.000000     0.5  0.5
  3   H      0.000000    0.629312   -0.889981     0.5  0.5
  4   H      0.000000    0.629312    0.889981     0.5  0.5
  5   H      0.889981   -0.629312    0.000000     0.5  0.5
Atoms.SpeciesAndCoordinates>
```

where the sixth and seventh columns give the number of initial charges for up and down spin states for each atom, respectively. The sum of up and down charges for the atomic element should be equivalent to the number of electrons which is taken into account in the pseudopotential generation. Then, the proper number for each pseudopotential can be found in the pseudopotential file '*.vps'. For example, you will see the following line in the file 'C_PBE19.vps' for carbon atom in the database Ver. 2019.

```
 valence.electron            4.0000
```

The number '4.0' corresponds to the number of electrons which is taken into account in the pseudopotential generation. So, we see in the above example that the sum of up (2.0) and down (2.0) spins charges is 4.0 for 'C' in the specification of 'Atoms.SpeciesAndCoordinates'. In Tables 2 and 1 we show the number of valence electrons in the pseudopotentials provided as the database Ver. 2019.

When you make pseudopotentials using ADPACK by yourself, you should pay attention to the following points.

- Check whether unphysical calculations have been caused by the ghost states or not. Because of the use of the separable form, the ghost states often appear. You should check whether the pseudopotentials are appropriate or not by performing calculations of simple systems before you calculate systems that you are interested in.

- Make smooth core densities for the partial core correction. If not so, numerical instabilities appear often, since a high energy cutoff is needed for accurate numerical integrations.

You will find the further details in the manual of the program package 'ADPACK'. However, it is noted that generation of good pseudopotentials requires considerable experiences more than what we think at the beginning.

### 11.2 Open core pseudopotentials

The $4f$-states in lanthanide elements are spin-polarized in many cases, and the majority states are located at below a few eV taken from the Fermi level. However, LDAs and GGAs cannot describe the feature of band structures for those materials. Although one way is to perform the plus U method by introducing on-site Coulomb repulsion for the $4f$-states in such a case, OpenMX provides a simpler way that the spin-polarizaion of 4f-states is taken into account via a pseudopotential, so-called *open core* pseudopotential, while a few open core pseudopotentials are available in the database Ver. 2019. The open core pseudopotential of a lanthanide element is generated by assuming that the 4f-states are a part of core states, and the partial core correction charge is generated so that the radial shape resembles well the charge distribution of the 4f-states. Pseudopotentials: Nd_CA19_OC.vps and Nd_PBE19_OC.vps stored in the database Ver. 2019 were generated in this way.

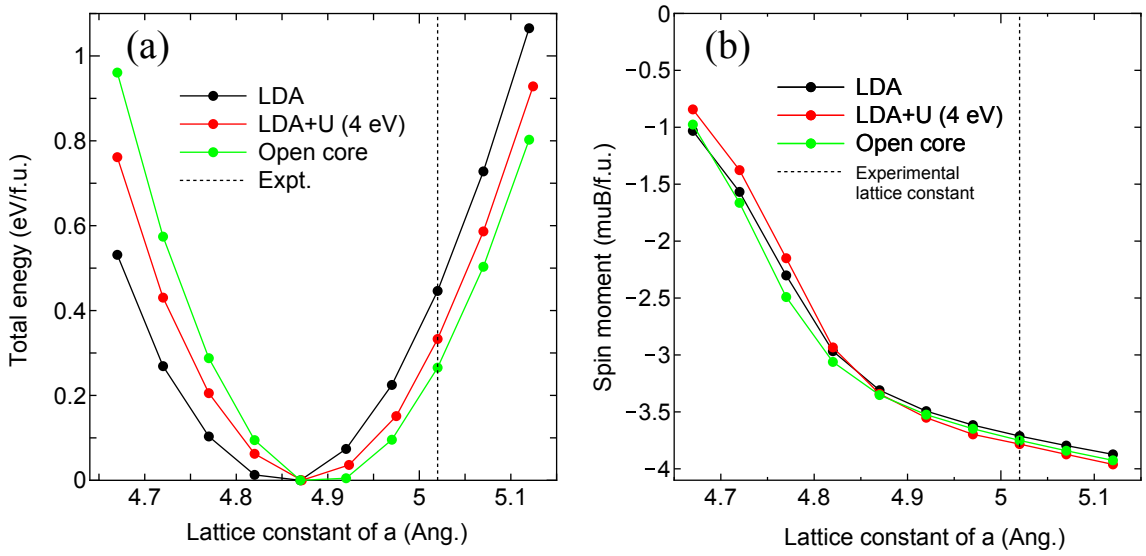



Figure 4: (a) The total energy of $NdCo_5$ in the $CaCu_5$ structure as a function of lattice constant, calculated by the LDA, LDA+U (U=4 eV for 4f-states), and open core pseudopotential methods. (b) Spin magnetic moment per the formula unit of the same system as (a). The input files used for the calculations are 'NdCo5_4f.dat', 'NdCo5_4f+U.dat', and 'NdCo5_OC.dat', which can be found in the directory 'work'.

To illustrate how the open core pseudopotential can be used, a series of calculations for $NdCo_5$ in the $CaCu_5$ structure is shown in Fig. 4. It is found that the calculation with the open core pseudopotential qualitatively reproduces the result by the LDA+U method. When the open core pseudopotential is used by OpenMX, the partial core correction charge can be spin-polarized by the following keyword:

```
<scf.pcc.opencore
  Nd  1
  Co  0
scf.pcc.opencore>
```

The example is for the $NdCo_5$ calculation discussed above. The first column is the name of species which is defined by the keyword 'Definition.of.Atomic.Species', and the second is a flag to specify the spin direction, where '1' and '-1' mean that the partial core correction charge is fully spin-polarized upward and downward along the $z$-axis, respectively, and '0' means no spin-polarization. Using the keyword, one can control the spin direction of $4f$ states being open core states site by site. It is also noted that the open core pseudopotential is valid if the overlap between the $4f$-states and orbitals in the neighboring atoms is negligible, and if the occupation of the $4f$-states is not largely different between the pseudopotential generation for an atom and the states in compounds.

### 11.3 Pseudopotentials for core level excitations

The database of fully relativistic pseudopotentials (VPS) and pseudo-atomic orbitals (PAO), which can be used for calculations of core level excitations, are available as Database (2019) for core excitations at the following website:

```
https://t-ozaki.issp.u-tokyo.ac.jp/vps_pao_core2019/
```

The data for B, C, N, O, Si, S, Ge, Pt elements are available. When you calculate absolute binding energies of core levels in bulk and gaseous systems, which can be measured in X-ray photoemission spectroscopy (XPS), the VPS and PAO files have to be used. See also the section 'Absolute binding energies of core levels: XPS core level energies'.

# 12 Cutoff energy: grid fineness for numerical integrations

## 12.1 Convergence

The computational effort and accuracy depend on the cutoff energy, which is controlled by the keyword 'scf.energycutoff', for the numerical integrations and the solution of Poisson's equation [42]. Figure 5 shows the convergence of the total energy of a methane molecule with respect to the cutoff energy, where the input file is 'Methane.dat' used in the Section 'Input file'. Since the cutoff energy is not for basis set as in plane wave methods, but for the numerical integrations, the total energy does not have to converge from the upper energy region with respect to the cutoff energy like that of plane wave basis set. In most cases, the cutoff energy of 150-200 Ryd is an optimum choice. However, it should be noted that there is a subtle problem which requires the cutoff energy more than 300 Ryd. Calculations of a very flat potential minimum and a small energy difference among different spin orders could be such a subtle problem.

Structural parameters and the dipole moment of a water molecule, calculated with a different cutoff energy, are shown in Table 3, where the input file is 'H2O.dat' in the directory 'work'. A convergent result is obtained using around 90 Ryd. Although a sufficient cutoff energy depends on elements, 150-200 Ryd might be enough to achieve the convergence for most cases. However, we recommend that you would check physical properties for your system. For the other cutoff energy, 1DFFT.EnergyCutoff, we commonly use 3600 (Ryd) which is quite enough for the convergence with no high computational demands.

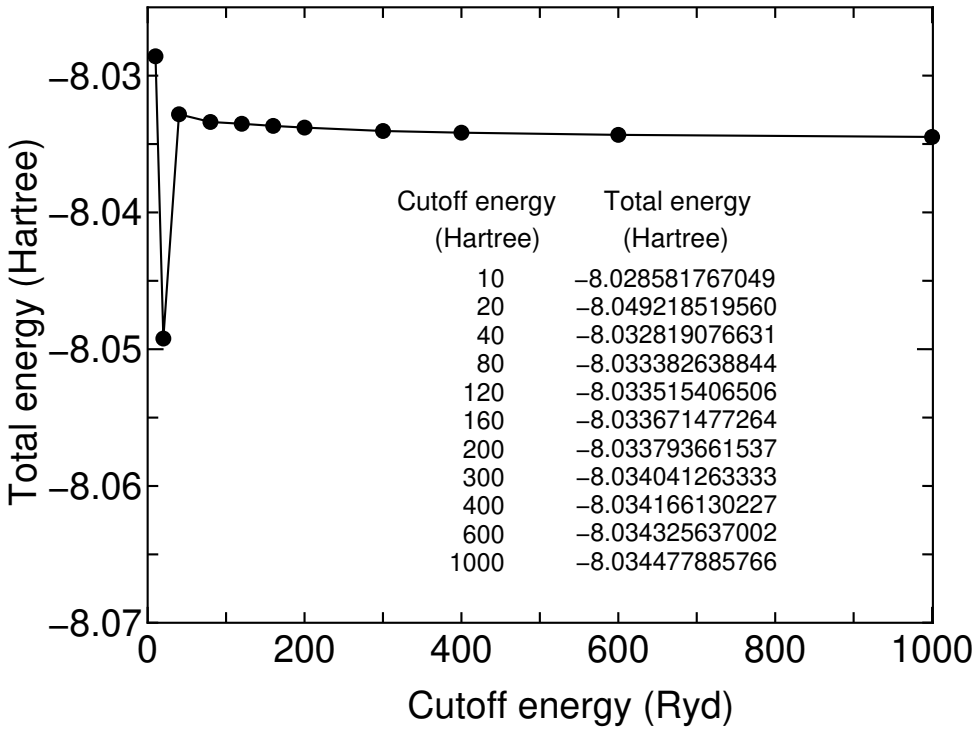



Figure 5: Convergence of the total energy of a methane molecule with respect to the cutoff energy.

Table 3: Convergence of structural parameters, dipole moment of a water molecule with respect to the cutoff energy. The input file is 'H2O.dat' in the directory 'work'.

| Ecut(Ryd) | r(H-O) (Å) | ∠ (H-O-H) (deg) | Dipole moment (Debye) |
|---|---|---|---|
| 60 | 0.970 | 103.4 | 1.838 |
| 90 | 0.971 | 103.7 | 1.829 |
| 120 | 0.971 | 103.7 | 1.832 |
| 150 | 0.971 | 103.6 | 1.829 |
| 180 | 0.971 | 103.6 | 1.833 |
| Exp. | 0.957 | 104.5 | 1.85 |

### 12.2 A tip for calculating the energy curve for bulks

When the energy curve for bulk system is calculated as a function of the lattice parameter, a sudden change of the number of real space grids is a serious problem which produces an erratic discontinuity on the energy curve. In fact, we see the discontinuity in cases of 200 and 290 (Ryd) in Fig. 6 when the cutoff energy is fixed. The discontinuity occurs at the lattice parameter where the number of grids changes. To avoid the discontinuity on the energy curve, a keyword 'scf.Ngrid' is available.

```
scf.Ngrid     32 32 32    # n1, n2, and n3 for a-, b-, and c-axes
```

When the number of grids is explicitly specified by the keyword, the axis is discretized by the number without depending on the keyword 'scf.energycutoff'. We see in Fig. 6 that the fixed grids with $32 \times 32 \times 32$ gives a smooth curve, while the discontinuity is not so serious even in the cases of 'scf.energycutoff'.

### 12.3 Fixing the relative position of regular grid

OpenMX Ver. 3.9 uses the regular real space grid for the evaluation of Hamiltonian matrix elements associated with the Hartree potential by the difference charge density and exchange-correlation potential, and solution of Poisson's equation. Thus, the total energy depends on the relative position between atomic coordinates and the regular grid. When one calculates an interaction energy or energy curve as a function of atomic coordinates, it is quite important to keep the relative position in all the calculations required for the evaluation of the interaction energy. In the calculation for one of the structures, you will find 'Grid_Origin' in the standard output which gives $x$-, $y$-, and $z$-components of the origin of the regular grid as:

```
Grid_Origin  xxx  yyy  zzz
```

Then, in order to keep the relative position, you have to include the following keyword 'scf.fixed.grid' in your input file for all the systems in the calculations required for the evaluation of the interaction energy:

```
scf.fixed.grid  xxx  yyy  zzz
```

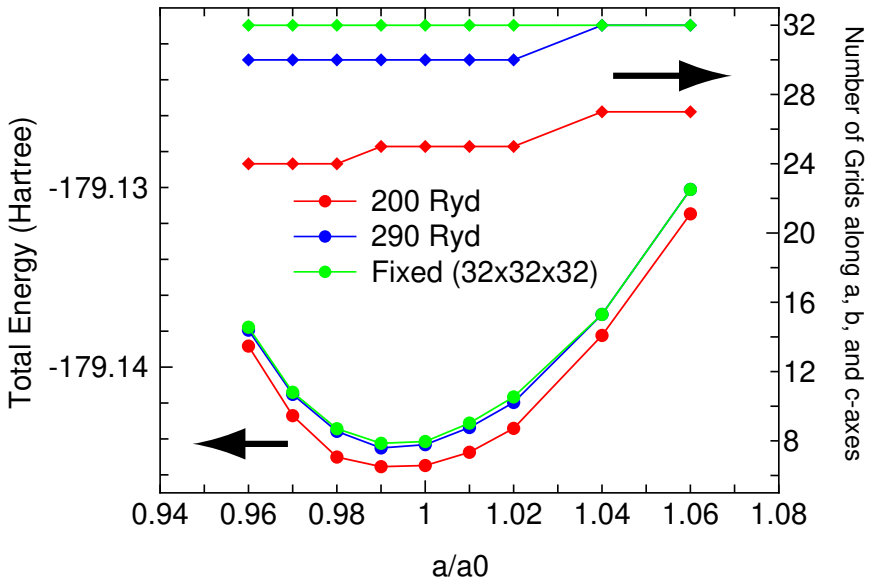



Figure 6: The total energy of bcc iron as a function of the lattice parameter, where the experimental equilibrium lattice constant $a_0$ is 2.87 Å. A cubic unit cell including two atoms was considered. The input file is 'Febcc2.dat' in the directory 'work'.

where 'xxx yyy zzz' is the coordinate of the origin you got in the calculation for one of the structures. The procedure largely suppresses the numerical error involved in the use of the regular grid. In addition, as discussed in the previous subsection 'A tip for calculating the energy curve for bulks', the number of grids should be fixed by the keyword 'scf.Ngrid' when the lattice parameters are also changed in the evaluation of interaction energy.

# 13 SCF convergence

## 13.1 General

Seven charge mixing schemes in OpenMX Ver. 3.9 are available by the keyword 'scf.Mixing.Type':

- Simple mixing (Simple)

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight

- Residual minimization method in the direct inversion iterative subspace (RMM-DIIS) [58]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Mixing.History, scf.Mixing.StartPulay

- Guaranteed reduction Pulay method (GR-Pulay) [57]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Mixing.History, scf.Mixing.StartPulay

- Kerker mixing (Kerker) [59]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Kerker.factor

- RMM-DIIS with Kerker metric (RMM-DIISK) [58]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Mixing.History, scf.Mixing.StartPulay, scf.Mixing.EveryPulay, scf.Kerker.factor

- RMM-DIIS for Kohn-Sham potentials with Kerker metric (RMM-DIISV) [58]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Mixing.History, scf.Mixing.StartPulay, scf.Mixing.EveryPulay, scf.Kerker.factor

- RMM-DIIS for Kohn-Sham Hamiltonian (RMM-DIISH) [58]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Mixing.History, scf.Mixing.StartPulay, scf.Mixing.EveryPulay,

In the first three schemes density matrices, which are regarded as a quantity in real space, are mixed to generate the input density matrix which can be easily converted into (spin) charge density. On the other hand, the charge mixing is made in Fourier space in the next three schemes: 'Kerker', 'RMM-DIISK', and 'RMM-DIISV'. The last scheme, 'RMM-DIISH', mixes Kohn-Sham Hamiltonian matrices, which may be suitable for the plus U method and the constraint schemes. Generally, it is easier to achieve SCF convergence in large gap systems using any mixing scheme. However, it would be difficult to achieve a sufficient SCF convergence in smaller gap and metallic systems, since a charge sloshing problem in the SCF calculations becomes serious often. To handle such difficult systems, three mixing schemes are currently available: 'Kerker', 'RMM-DIISK', and 'RMM-DIISV' methods. The three mixing schemes could be an effective way for achieving the SCF convergence of metallic systems. When 'Kerker', 'RMM-DIISK', or 'RMM-DIISV' is used, the following prescriptions are helpful to obtain the convergence of SCF calculations:

- Increase of 'scf.Mixing.History'. A relatively larger vaule 30-50 may lead to the convergence. In addition, 'scf.Mixing.EveryPulay' should be set in 1.
- Use a rather larger value for 'scf.Mixing.StartPulay'. Before starting the Pulay-type mixing, achieve a convergence at some level. An appropriate value may be 10 to 30 for 'scf.Mixing.StartPulay'.
- Use a rather larger value for 'scf.ElectronicTemperature' in case of metallic systems. When 'scf.ElectronicTemperature' is too low, numerical instabilities appear often.

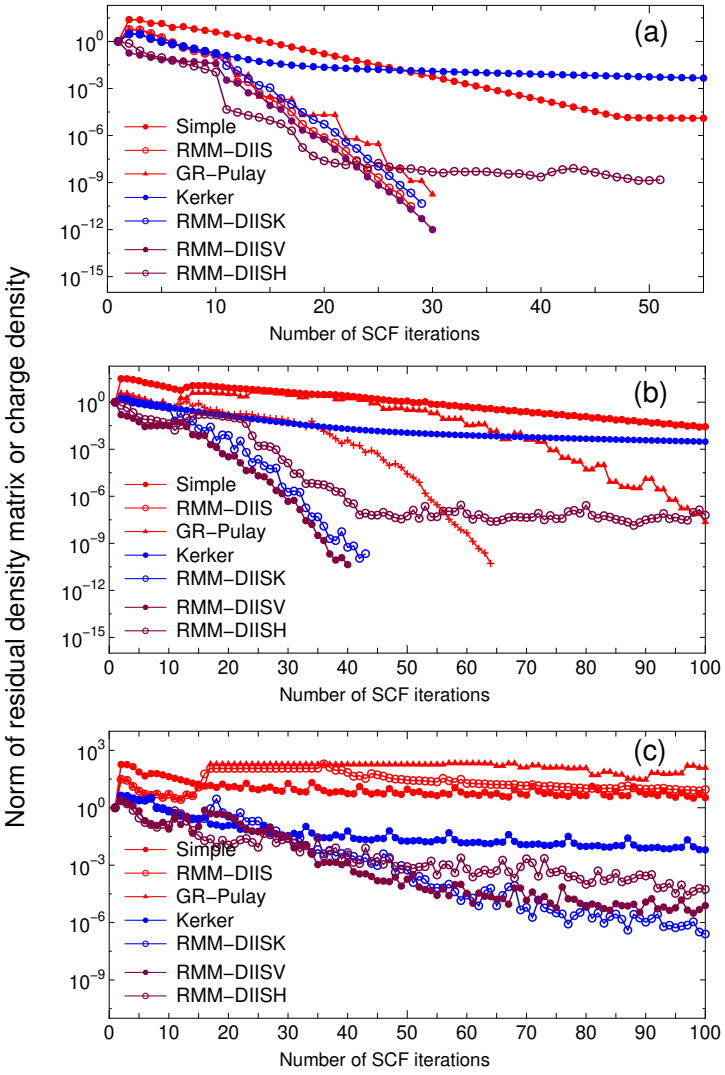



Figure 7: Convergence of the norm of residual density matrix or charge density in the SCF calculations using five mixing schemes of (a) a sialic acid molecule, (b) a $Pt_{13}$ cluster, and (c) a $Pt_{63}$ cluster. The input files are 'SialicAcid.dat', 'Pt13.dat', and 'Pt63.dat' in the directory 'work'.

In addition, the charge sloshing, which comes from charge components with long wave length, can be significantly suppressed by tuning Kerker's factor $\alpha$ by the keyword 'scf.Kerker.factor', where Kerker's

metric is defined by

$$\langle A|B\rangle = \sum_{\mathbf{q}} \frac{A^*_{\mathbf{q}} B_{\mathbf{q}}}{w_{\mathbf{q}}}$$

$$w_{\mathbf{q}} = \frac{|\mathbf{q}|^2}{|\mathbf{q}|^2 + q_0^2}$$

$$q_0 = \alpha |\mathbf{q}_{\min}|$$

where $\mathbf{q}_{\min}$ is the $\mathbf{q}$ vector with the minimum magnitude except 0-vector. A larger $\alpha$ significantly suppresses the charge sloshing, but leads to slower convergence. Since an optimum value depends on system, you may tune an appropriate value for your system.

Furthermore, the behavior of 'RMM-DIISK' can be controlled by the following keyword:

```
scf.Mixing.EveryPulay    5   # default = 1
```

The residual vectors in the Pulay-type mixing schemes tend to become linearly dependent on each other as the mixing steps accumulate, and the linear dependence among the residual vectors makes the convergence difficult. A way of avoiding the linear dependence is to do the Pulay-type mixing occasionally during the Kerker mixing. With this prescription, you can specify the frequency using the keyword 'scf.Mixing.EveryPulay'. For example, in case of 'scf.Mixing.EveryPulay=5', the Pulay-mixing is made at every five SCF iterations, while the Kerker-type mixing is used at the other steps. 'scf.Mixing.EveryPulay=1' corresponds to the conventional Pulay-type mixing. It is noted that the keyword 'scf.Mixing.EveryPulay' is supported for only 'RMM-DIISK', and the default value is '1'.

The above prescription works in some cases. But the most recommended prescription to accelerate the convergence is the following:

- Increase of 'scf.Mixing.History'. A relatively larger vaule 30-50 may lead to the convergence. In addition, 'scf.Mixing.EveryPulay' should be set in 1.

Since the Pulay-type mixing such as 'RMM-DIIS', 'RMM-DIISK', and 'RMM-DIISV' is based on a quasi Newton method, the convergence speed is governed by how a good approximate Hessian matrix can be found. As 'scf.Mixing.History' increases, the calculated Hessian may become more accurate.

In Fig. 7 a comparison of seven mixing schemes is shown for the SCF convergence for (a) a sialic acid molecule, (b) a $Pt_{13}$ cluster, and (c) a $Pt_{63}$ cluster, where the norm of residual density matrix or charge density can be found as NormRD in the file '*System.Name*.out' and the input files are 'SialicAcid.dat', 'Pt13.dat', and 'Pt63.dat' in the directory 'work'. We see that 'RMM-DIISK' and 'RMM-DIISV' work with robustness for all the systems shown in Fig. 7. In most cases, 'RMM-DIISK' and 'RMM-DIISV' will be the best choice, while the use of 'Kerker' is required with a large 'scf.Kerker.factor' and a small 'scf.Max.Mixing.Weight' for quite difficult cases in which the convergence is hardly obtained. Also our experiences imply that 'RMM-DIISH' is suitable for the plus U method and the constraint schemes, while such a case is not shown in Fig. 7.

### 13.2 Automatic determination of Kerker's factor

If the keyword 'scf.Kerker.factor' is not given in your input file, OpenMX Ver. 3.9 automatically estimates a proper value of Kerker's factor $\alpha$ by the following equation:

$$\alpha = \frac{0.5}{|\mathbf{b}_{\rm min}|^2}\left(4\frac{Dq}{Aq}+1.0\right)$$

with

$$Aq = \frac{1}{3}\left(|\mathbf{b}_1|^2+|\mathbf{b}_2|^2+|\mathbf{b}_3|^2\right),$$

$$Dq = \frac{1}{3}\sum_{i<j}\left||\mathbf{b}_i|^2-|\mathbf{b}_j|^2\right|,$$

where $\mathbf{b}_i (i=1,2,3)$ is a reciprocal vector, and $\mathbf{b}_{\rm min}$ is the smallest vector among $\{\mathbf{b}\}$. The equation takes account of the dependency of $\alpha$ on the size and anisotropy of the system. From a series of numerical calculations it is found that the estimated value works well in most cases.

### 13.3 On-the-fly control of SCF mixing parameters

During the SCF calculation, it is possible to change the following parameters for the SCF mixing:

```
scf.maxIter
scf.Min.Mixing.Weight
scf.Max.Mixing.Weight
scf.Kerker.factor
scf.Mixing.StartPulay
```

For example, when you specify the following two keywords in your input file as

```
System.CurrrentDirectory          ./    # default=./
System.Name                       c60
```

then make a file whose name is 'c60_SCF_keywords' in the directory './', and write in it as

```
scf.maxIter               100
scf.Min.Mixing.Weight     0.01
scf.Max.Mixing.Weight     0.10
scf.Kerker.factor         10.0
scf.Mixing.StartPulay     30
scf.criterion             1.0e-6
```

OpenMX will try to read the file 'c60_SCF_keywords' at every SCF step, and show the following message in the standard output, if the file is successfully read by OpenMX.

```
The keywords for SCF iteration are renewed by ./c60_SCF_keywords.
```

Also, if a minus value is given for the keyword 'scf.maxIter', then OpenMX will be terminated. The on-the-fly control of SCF mixing parameters may be useful when large-scale calculations are performed.

# 14 Restarting

## 14.1 General

After finishing your first calculation or achieving the self consistency, you may want to continue the calculation or to calculate density of states, band dispersion, molecular orbitals, and etc. using the self consistent charge density in order to save the computational time. To do this, a keyword 'scf.restart' is available.

```
scf.restart            on        # on|off,default=off
```

When the keyword 'scf.restart' is switched on, restart files generated by your first calculation will be used as the input Hamiltonian or charge density in the second calculation, while 'System.Name' in the second calculation should be the same as in the first calculation. The restart files are stored in a directory '*System.Name*_rst' below the directory 'work', where *System.Name* means 'System.Name'. The restart files in the '*System.Name*_rst' contain all the information for both the density matrix mixing schemes and k-space mixing schemes. So, it is also possible to use another mixing scheme in the second calculation. As an example, we illustrate the restarting procedure using an input file *C60.dat* which can be found in the directory 'work'. In Fig. 8, we see that the second calculation is accelerated due to the use of the restart file.

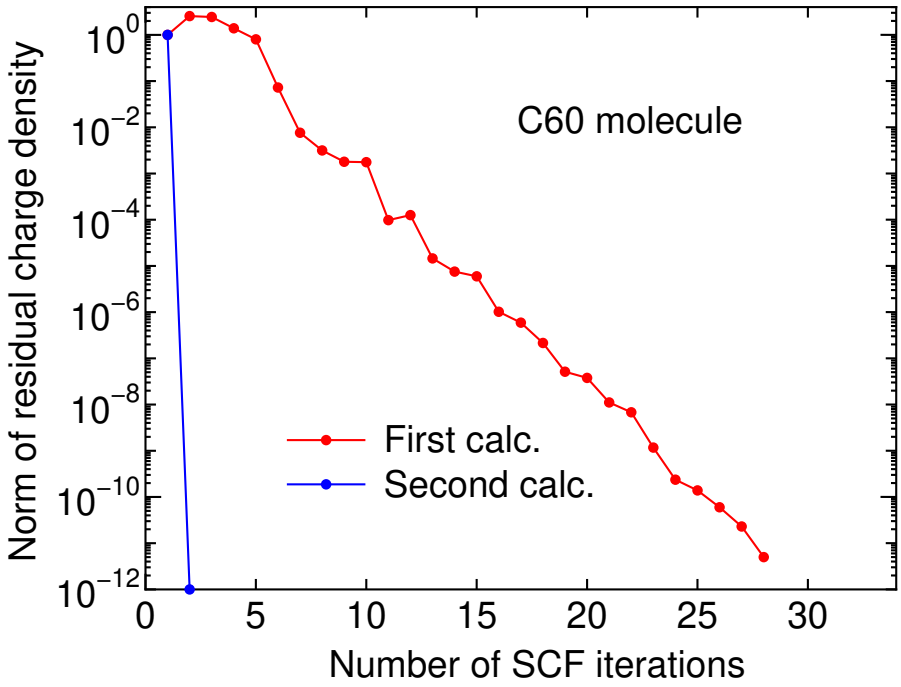



Figure 8: SCF convergence of a $C_{60}$ molecule. In the second calculation, the restart files generated by the first calculation were used. The input file is 'C60.dat' in the directory 'work'.

## 14.2 Extrapolation scheme during MD and geometry optimization

In the geometry optimization and molecular dynamics simulations, the restart files generated at the previous steps are automatically utilized at the next step to accelerate the convergence using an extrapolation scheme [60, 61]. In the extrapolation scheme, the number of previous MD or geometry optimization steps can be controlled by a keyword:

```
scf.ExtCharge.History      2        # default=2
```

From a series of benchmark calculations, 'scf.ExtCharge.History' of 2 works well and a larger number tends to be numerically unstable. So, we recommend for users to use the default setting of 2.

## 14.3 Input file for the restart calculation

An input file '*System.Name*.dat#' is generated at every MD step for the restart calculation with the final structure and the same 'Grid_Origin' explained in the Section 'Fixing the relative position of regular grid'. Using the file '*System.Name*.dat#', it can be possible to continue MD calculations and geometry optimization from the last step.

# 15 Geometry optimization

## 15.1 Steepest decent optimization

An example of the geometry optimization is illustrated in this Section. As an initial structure, let us consider the methane molecule given in the Section 'Input file', but the $x$-coordinate of the carbon atom of the methane molecule is moved to 0.3 Å as follows:

```
<Atoms.SpeciesAndCoordinates
  1   C      0.300000     0.000000     0.000000      2.0  2.0
  2   H     -0.889981    -0.629312     0.000000      0.5  0.5
  3   H      0.000000     0.629312    -0.889981      0.5  0.5
  4   H      0.000000     0.629312     0.889981      0.5  0.5
  5   H      0.889981    -0.629312     0.000000      0.5  0.5
Atoms.SpeciesAndCoordinates>
```

Then, a keyword 'MD.type' is specified as 'Opt', and set to 200 for a keyword 'MD.maxIter'. The 'Opt' is based on a simple steepest decent method with a variable prefactor. Figure 8 (a) shows the convergence history of the norm of the maximum force on atom as a function of the number of the optimization steps. We see that the norm of the maximum force on atom converges after the structure overshot the stationary point because of change of the prefactor. Using 'Methane2.dat' in the directory 'work', you can trace the calculation. As well as the case of the methane molecule, a similar behavior can be seen for the silicon diamond as shown in Fig. 9(b).

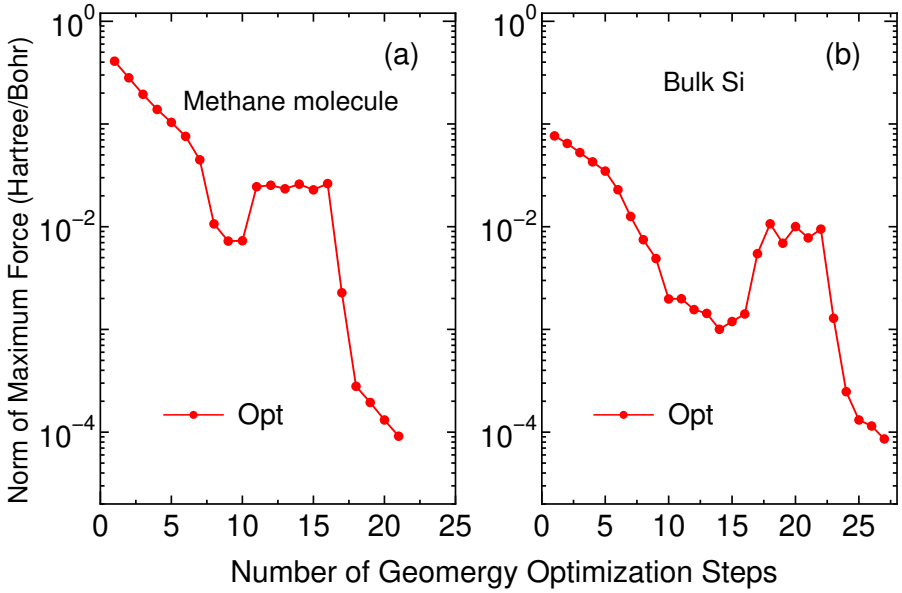



Figure 9: The norm of the maximum force on atom of (a) a methane molecule (b) silicon in the diamond structure as a function of geometry optimization steps. The initial structures are ones distorted from the the equilibrium structures. The input files are 'Methane2.dat' and 'Si8.dat' in the directory 'work', respectively.

## 15.2 EF, BFGS, RF, and DIIS optimizations

Although 'Opt' is a robust scheme, the convergence speed can be slow in general. Faster schemes based on quasi Newton methods are available for the geometry optimization. They are the eigenvector following (EF) method [63], the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [65], the rational function (RF) method [64], and a direct inversion iterative sub-space (DIIS) method [62], implemented in Cartesian coordinate. In the EF and RF methods, the approximate Hessian is updated by the BFGS method. Thus, five geometry optimizers, Opt, EF, BFGS, RF and DIIS, are available in OpenMX Ver. 3.9, which can be specified by 'MD.Type'. The relevant keywords are listed below:

```
MD.Type                    EF         # Opt|DIIS|BFGS|RF|EF
MD.Opt.DIIS.History         3         # default=3
MD.Opt.StartDIIS            5         # default=5
MD.Opt.EveryDIIS          200         # default=200
MD.maxIter                100         # default=1
MD.Opt.criterion        1.0e-4        # default=0.0003 (Hartree/Bohr)
```

Especially, you can control these schemes by two keywords:

```
MD.Opt.DIIS.History     3       # default=3
MD.Opt.StartDIIS        5       # default=5
```

The keyword 'MD.Opt.DIIS.History' specifies the number of the previous steps to update an optimum Hessian matrix. The default value is 3. Also, the geometry optimization step at which 'EF', 'BFGS', 'RF', or 'DIIS' starts is specified by the keyword 'MD.Opt.StartDIIS'. The geometry optimization steps before starting these methods is performed by the steepest decent method as in 'Opt'. The default value is 5.

The initial step in the optimization is automatically tuned by monitoring the maximum force in the initial structure. As shown in Fig. 10 which shows the number of geometry steps to achieve the maximum force of below 0.0003 Hartree/Bohr in molecules and bulks, in most cases the RF method seems to be the most robust and efficient scheme, while the EF and BFGS methods also show a similar performance. The input files used for those calculations and the out files can be found in the directory 'work/geoopt_example/'.

It should be also noted that by these quasi Newton methods geometrical structures tend to be converged to a saddle point rather than a stationary minimum point. This is because the structure at which the quasi Newton method started to be employed does not reach at a flexion point. In such a case, the structure should be optimized well by the steepest decent method before moving to the quasi Newton method. The treatment can be easily done by only taking a larger value for 'MD.Opt.StartDIIS', or by restarting the calculation using a file '*System.Name*.dat#', where '*System.Name*' is 'System.Name' specified in your input file.

In general, a faster convergence can be obtained by employing a large 'scf.energycutoff' leading to a smooth energy curve. This situation is apparent especially for weakly interacting systems such as molecular solids. We recommend for users to employ a large 'scf.energycutoff', e.g., 300-400 Ryd for such a system.

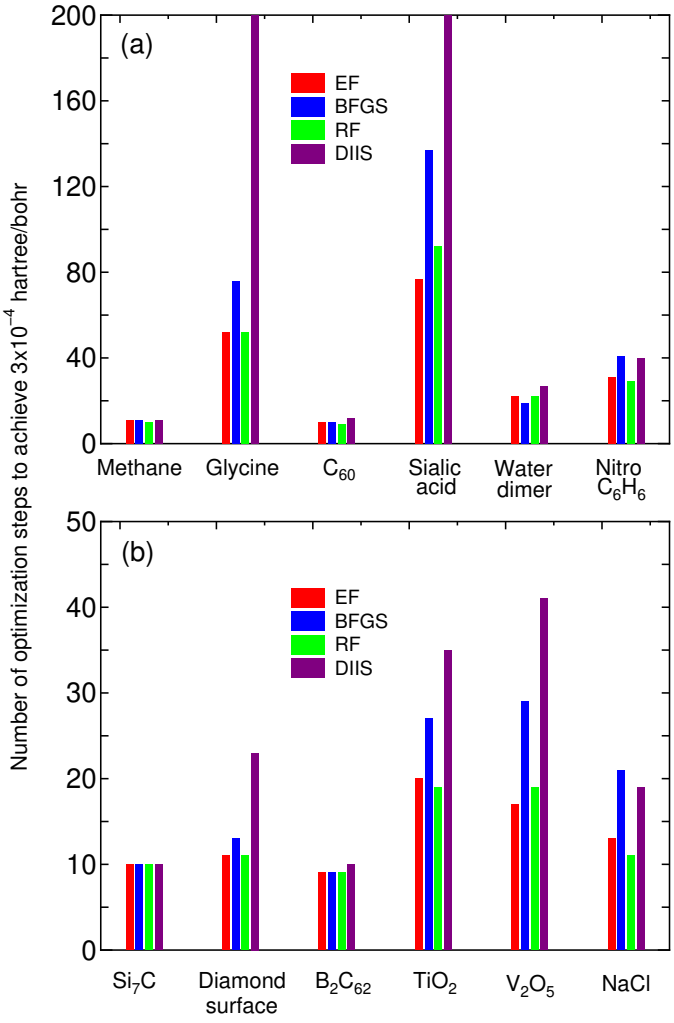



Figure 10: The number of optimization steps to achieve the maximum force of below $3 \times 10^{-4}$ Hartree/Bohr for (a) molecular systems and (b) bulk systems using four kinds of optimization methods.

### 15.3 Initial Hessian for the RF and EF optimizers

Two sorts of the initial approximate Hessian for the RF and EF optimizers in the geometry optimization are available in OpenMX Ver. 3.9, which can be specified by the following keyword:

```
MD.Opt.Init.Hessian         Schlegel      # Schlegel|iden, default=Schlegel
```

The defaut is 'Schlegel' which was proposed by Schlegel [66], and estimates the initial Hessian using a simple model consisting of bond stretching, angle bending, and torsion, while only the bond stretching term is taken into account in the implementation of OpenMX Ver. 3.9. The other is 'iden' which starts from the identity matrix for the approximate Hessian. In both the cases, the initial Hessian is updated every geometry optimization step by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [65]. It is noted that 'Schlegel' is not supported for the optimizers of 'BFGS' and 'DIIS'. In general, the method of Schlegel provides a faster convergence in the optimization compared to the initial approximate Hessian of the identity matrix. Thus, one should use 'Schlegel' being default as a first choice. In

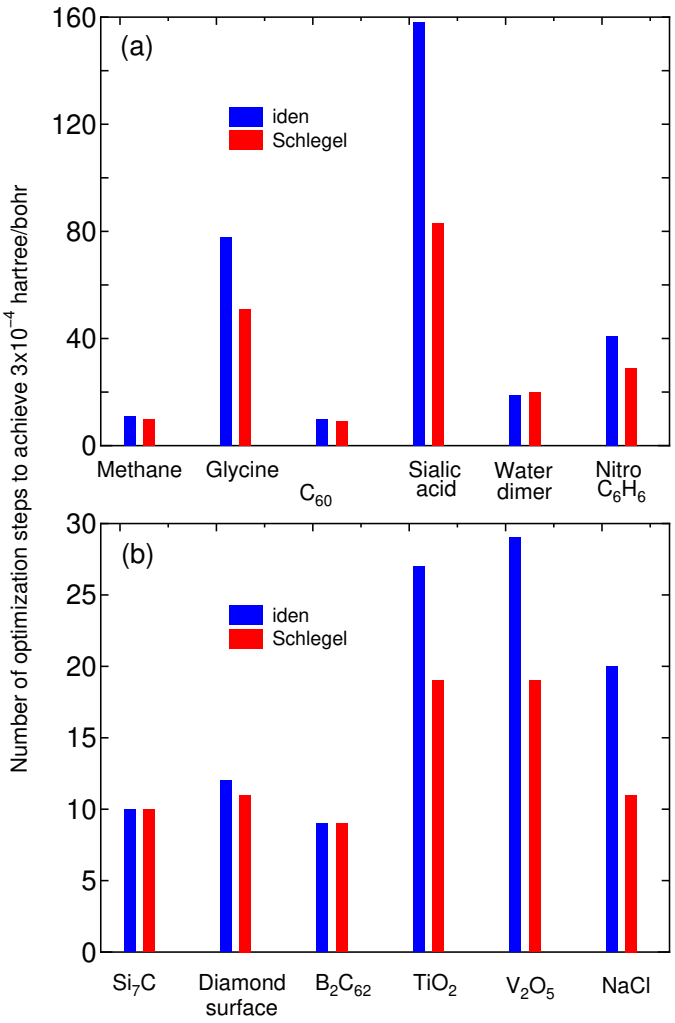



Figure 11: The number of optimization steps to achieve the maximum force of below $3 \times 10^{-4}$ Hartree/Bohr for (a) molecular systems and (b) bulk systems using the RF method with the initial Hessian of 'iden' and 'Schlegel'.

Fig. 11 a comparison between 'Schlegel' and 'iden' is shown during the geometry optimization using the EF method. It can be seen that the EF method with 'Schlegel' is faster than 'iden' in most cases including molecules and bulks.

### 15.4 Constrained relaxation

It is possible to optimize geometrical structures with a constraint in which atoms can be fixed in the initial position. The constraint can be applied separately to the $x$-, $y$-, and $z$-coordinates to the initial atomic position in your input file by the following keyword 'MD.Fixed.XYZ':

```
<MD.Fixed.XYZ
   1  1 1 1
   2  1 0 0
MD.Fixed.XYZ>
```

The example is for a system consisting of two atoms. If you have $N$ atoms, then you have to provide $N$ rows in this specification. The 1st column is the same sequential number to specify atom as in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. The 2nd, 3rd, and 4th columns are flags for the $x$-, $y$-, and $z$-coordinates, respectively. '1' means that the coordinate is fixed, and '0' relaxed. In the above example, the $x$-, $y$-, and $z$-coordinates of the atom '1' are fixed, and only the x-coordinate of the atom '2' is fixed. The default setting is that all the coordinates are relaxed. The fixing of atomic positions are valid for all the geometry optimizers and molecular dynamics schemes. The constrained relaxation may be useful for a refinement of the local structure in large-scale systems.

### 15.5 Restart of geometry optimization

If the first trial for geometry optimization does not reach a convergent result, one can restart the geometry optimization using an input file '*System.Name*.dat#' which is generated at every geometry optimization step for the restart calculation with the final structure. In such a case, it is better to restart the optimization with the approximate Hessian matrix calculated in the first trial to accelerate the convergence. In OpenMX Ver. 3.9, the approximate Hessian matrix is also saved every geometry optimization step, and is reused when the restart is performed by '*System.Name*.dat#'. Thus, even if the geometry optimization is intermittently repeated by subsequent job submission, the number of iterations for the geometry optimization step is the same as that in the single submission. The functionality may be useful when users optimize large-scale systems using computational systems in common use for which the wall time is set for each job.

# 16 Variable cell optimization

## 16.1 General

The variable cell optimizations with/without constraints are supported in OpenMX Ver. 3.9. The relevant keywords for the variable cell optimizations are listed below:

```
MD.Type                    RFC5       # OptC1|OptC2|OptC3|OptC4|OptC5
                                      # OptC6|OptC7|RFC5|RFC6|RFC7
MD.Opt.DIIS.History         3         # default=3
MD.Opt.StartDIIS            5         # default=5
MD.Opt.EveryDIIS          200         # default=200
MD.maxIter                100         # default=1
MD.Opt.criterion        1.0e-4        # default=0.0003 (Hartree/Bohr)
```

As confirmed, the keywords listed above are exactly the same as in the section 'Geometry optimization'. Thus, the variable cell optimization can be controlled just like the geometry optimization. The variable cell optimization is supported for only the collinear calculations including the plus U method, while, however, the cell optimization for the DFT-D2 and DFT-D3 methods for vdW interaction is not supported. By the keyword 'MD.Type', a method for the variable cell optimization is specified. When you perform the variable cell optimization, you can choose the following option for the keyword 'MD.Type':

- OptC1

  Cell vectors are optimized without any constraint, while keeping the initial fractional coordinates. The optimization is performed by a steepest decent method with a variable prefactor.

- OptC2

  Cell vectors are optimized with a constraint that angles between cell vectors are fixed at the initial values, while keeping the initial fractional coordinates. Thus, only the length of cell vectors is optimized during the optimization. The optimization is performed by a steepest decent method with a variable prefactor.

- OptC3

  Cell vectors are optimized with a constraint that angles between cell vectors are fixed at the initial values and the length of cell vectors is equivalent to each other: $|\mathbf{a}_1| = |\mathbf{a}_2| = |\mathbf{a}_3|$, while keeping the initial fractional coordinates during the optimization. Thus, only the length of cell vectors is optimized. The optimization is performed by a steepest decent method with a variable prefactor.

- OptC4

  Cell vectors are optimized with a constraint that angles between cell vectors are fixed at the initial values and the length of cell vectors is optimized under a condition: $|\mathbf{a}_1| = |\mathbf{a}_2| \neq |\mathbf{a}_3|$, while keeping the initial fractional coordinates. Thus, only the length of cell vectors is optimized during the optimization. The optimization is performed by a steepest decent method with a variable prefactor.

- OptC5

  Cell vectors and internal coordinates are simultaneously optimized without any constraint by using a steepest decent method with a variable prefactor.

- OptC6

  Cell vectors and internal coordinates are simultaneously optimized with a constraint that a cell vector $\mathbf{a}_3$ is fixed. The optimization is performed with a steepest decent method with a variable prefactor.

- OptC7

  Cell vectors and internal coordinates are simultaneously optimized with a constraint that two cell vectors $\mathbf{a}_2$ and $\mathbf{a}_3$ are fixed. The optimization is performed with a steepest decent method with a variable prefactor.

- RFC5

  Cell vectors and internal coordinates are simultaneously optimized without any constraint by using a combination scheme of the rational function (RF) method [64] and the direct inversion iterative sub-space (DIIS) method [62] with a BFGS update [65] for the approximate Hessian. The initial Hessian is given by an identity matrix or a model Hessian by Schlegel [66], which can be specified by the keyword 'MD.Opt.Init.Hessian' in the same way as in the geometry optimization. See the details for the section 'Geometry optimization'.

- RFC6

  Cell vectors and internal coordinates are simultaneously optimized witht a constraint that a cell vector $\mathbf{a}_3$ is fixed. The optimization is performed with the same way as for the RFC5.

- RFC7

  Cell vectors and internal coordinates are simultaneously optimized witht a constraint that two cell vectors $\mathbf{a}_2$ and $\mathbf{a}_3$ are fixed. The optimization is performed with the same way as for the RFC5.

Depending on your purpose, one of the options listed above should be properly chosen. Other constraint schemes will be implemented in future work.

As an example of the variable cell optimization, we show the simultaneous optimization of cell vectors and internal coordinates for the diamond primitive cell below. The calculation can be performed by

```
% mpirun -np 16 openmx Cdia-RFC5.dat > Cdia-RFC5.std &
```

where the input file 'Cdia-RFC5.dat' can be found in the directory 'work/cellopt_example', so that you can trace the same calculation. As an illustration the initial structure is distorted as shown below:

```
Atoms.Number         2
Atoms.SpeciesAndCoordinates.Unit   frac # Ang|AU
<Atoms.SpeciesAndCoordinates
   1    C    0.10000000000000    0.00000000000000   -0.05000000000000     2.0     2.0
   2    C    0.25000000000000    0.25000000000000    0.25000000000000     2.0     2.0
```

```
Atoms.SpeciesAndCoordinates>
Atoms.UnitVectors.Unit             Ang # Ang|AU
<Atoms.UnitVectors
  1.6400  1.6400  0.0000
  1.6400  0.0000  1.6400
  0.0000  1.6400  1.6400
Atoms.UnitVectors>
```

Using a cluster machine consisting of Intel Xeon of 2.6 GHz, the elapsed time of the calculation was 326 sec., which corresponds to 12 optimization steps. The history of the total energy and the maximum gradients of the total energy with respect to atomic coordinates or cell vectors can be confirmed in '*System.Name*.out'. You may find the following information in 'Cdia-RFC5.out' for the case.

```
*******************************************************
*******************************************************
               History of cell optimization
*******************************************************
*******************************************************
```

| MD_iter | SD_scaling | \|Maximum force\| (Hartree/Bohr) | Maximum step (Ang) | Utot (Hartree) | Enpy (Hartree) | Volume (Ang^3) |
|---|---|---|---|---|---|---|
| 1 | 1.25981732 | 0.16438857 | 0.10583545 | -11.59621750 | -11.59621750 | 8.82188800 |
| 2 | 1.25981732 | 0.08853079 | 0.05902053 | -11.64994330 | -11.64994330 | 9.81261691 |
| 3 | 1.25981732 | 0.04581932 | 0.03054622 | -11.66453803 | -11.66453803 | 10.28662955 |
| 4 | 1.25981732 | 0.02205340 | 0.01470227 | -11.66928384 | -11.66928384 | 10.56026328 |
| 5 | 3.14954331 | 0.01336972 | 0.02228286 | -11.67121215 | -11.67121215 | 10.73689973 |
| 6 | 3.14954331 | 0.00678359 | 0.01130598 | -11.67332696 | -11.67332696 | 11.04288573 |
| 7 | 3.14954331 | 0.00487464 | 0.01195765 | -11.67421713 | -11.67421713 | 11.13669753 |
| 8 | 3.14954331 | 0.00354039 | 0.02370087 | -11.67479906 | -11.67479906 | 11.18107598 |
| 9 | 3.14954331 | 0.00157491 | 0.00373195 | -11.67534267 | -11.67534267 | 11.29495641 |
| 10 | 3.14954331 | 0.00137813 | 0.00160469 | -11.67537385 | -11.67537385 | 11.34330266 |
| 11 | 3.14954331 | 0.00067979 | 0.00165878 | -11.67538616 | -11.67538616 | 11.37836604 |
| 12 | 3.14954331 | 0.00003708 | 0.00000000 | -11.67538985 | -11.67538985 | 11.39519327 |

It can be seen that the absolute value of the maximum gradient rapidly converged, and dropped to below the criterion of 0.0003 Hartree/bohr.

Other examples (input and output files) for the variable cell optimization can be found in a directory 'work/cellopt_example'.

## 16.2 Stress tensor

In the previous subsection, we discussed how the variable cell optimizations can be performed by specifying the keyword 'MD.Type'. In these cases, the stress tensor is automatically calculated by analytically evaluating the gradient of the total energy with respect to strain, and then transformed to the gradients with respect to cell vectors. If you do not perform the variable cell optimization, and however want to calculate the gradient of the total energy with respect to cell vectors, the following keyword is available:

```
scf.stress.tensor  on  # on|off, default=off
```

When the keyword 'scf.stress.tensor' is switched on, you may find the gradient of the total energy with respect cell vectors in '*System.Name*.out.

### 16.3 Constraint for cell vectors

When the options 'OptC1', 'OptC2', or 'OptC5' is used for the keyword 'MD.Type', each Cartesian component of cell vectors can be fixed at the initial value by using the following keyword :

```
<MD.Fixed.Cell.Vectors
 0 0 1
 0 0 0
 0 0 0
MD.Fixed.Cell.Vectors>
```

where the first line provides the flag for $a_x$, $a_y$, $a_z$, the second $b_x$, $b_y$, $b_z$, and the third $c_x$, $c_y$, $c_z$, respectively. '1' means that the component is fixed, and '0' relaxed.

### 16.4 Optimization of enthalpy

It is possible to perform the variable cell optimization under an applied pressure. This is done by minimizing the enthalpy $H$ defined with

$$H = E + pV, \tag{1}$$

where $E$ is the total energy of system per unit cell, $p$ is the applied pressure, and $V$ is the volume of the unit cell. To perform the optimization of enthalpy, you only have to include the following keyword in your input file:

```
MD.applied.pressure          10.0       # in GPa, default=0
```

The positive pressure corresponds to the compression of cell. The functionality is compatible with 'OptC1', 'OptC2', 'OptC3', 'OptC4', 'OptC5', 'OptC6', 'OptC7', and 'RFC5', 'RFC6', and 'RFC7' which can be specified by the keyword 'MD.Type'. As an example, one can perform the enthalpy optimization using an input file 'Si8-pV.dat' stored in the directory 'work', which is for the optimization of Si crystal under 10 GPa. After the calculation, you may find the history of the optimization in the out file 'si8-pV.out' as follows:

```
*******************************************************
               History of cell optimization
*******************************************************
*******************************************************
```

| MD_iter | SD_scaling | \|Maximum force\| (Hartree/Bohr) | Maximum step (Ang) | Utot (Hartree) | Enpy (Hartree) | Volume (Ang^3) |
|---|---|---|---|---|---|---|
| 1 | 1.25981732 | 0.07663140 | 0.05108760 | -32.84057849 | -32.47335956 | 160.10300700 |
| 2 | 1.25981732 | 0.06717954 | 0.04478636 | -32.84541333 | -32.48138995 | 158.70978745 |
| 3 | 1.25981732 | 0.05879663 | 0.03919775 | -32.84853574 | -32.48736913 | 157.46427382 |
| 4 | 1.25981732 | 0.05131728 | 0.03421152 | -32.85047522 | -32.49182806 | 156.36581813 |
| 5 | 3.14954331 | 0.04468030 | 0.07446716 | -32.85159836 | -32.49515060 | 155.40690918 |
| 6 | 3.14954331 | 0.02956430 | 0.04927383 | -32.85232293 | -32.50062291 | 153.33695214 |
| 7 | 3.14954331 | 0.01960389 | 0.03267316 | -32.85158714 | -32.50293764 | 152.00696345 |
| 8 | 3.14954331 | 0.01318467 | 0.02197446 | -32.85069024 | -32.50392104 | 151.18717226 |
| 9 | 7.87385828 | 0.00909382 | 0.03789092 | -32.84998761 | -32.50434789 | 150.69473500 |
| 10 | 7.87385828 | 0.00253118 | 0.00537839 | -32.84867882 | -32.50470324 | 149.96919000 |

```
11      7.87385828      0.00198428      0.03321825      -32.84877730      -32.50477155      149.98234416
12      7.87385828      0.00271856      0.01866538      -32.84922787      -32.50499775      150.08016284
13      7.87385828      0.00086782      0.00943670      -32.84942256      -32.50507226      150.13256385
14      7.87385828      0.00077020      0.00982293      -32.84949585      -32.50509162      150.15607426
15      7.87385828      0.00020223      0.00270074      -32.84950610      -32.50511244      150.15146767
16      7.87385828      0.00005544      0.00000000      -32.84950546      -32.50511390      150.15055140
```

It can be confirmed that the enthalpy is actually optimized with shrinking of the volume rather than the total energy.

Also, one can apply the pressure to only designated axes in orthorhombic crystal systems by the following keyword:

```
MD.applied.pressure.flag   1 1 1   # default=1 1 1
```

The default setting is '1 1 1' which means that the isotropic pressure is equally applied along the **a**-, **b**-, and **c**-axes. When you specify the keyword as '1 0 0', the pressure is applied along only the **a**-axis which is the direction perpendicular to the **bc**-plane in orthorhombic crystal systems. The functionality is valid for orthorhombic crystal systems, and in this case you need to provide the unit vectors such as

```
<Atoms.UnitVectors
 10.000   0.000   0.000
  0.000   8.000   0.000
  0.000   0.000  11.000
Atoms.UnitVectors>
```

so that the non-zero elements can be diagonal. Apart from the orthorhombic crystal systems, one may apply the anisotropic pressure along an lattice vector perpendicular to the plane defined by the other lattice vectors. The case can be found in cases such as hexagonal crystal systems, and you may specifiy these keywords as follows

```
MD.applied.pressure.flag   0 0 1   # default=1 1 1


<Atoms.UnitVectors
  6.000   0.000   0.000
 -3.000   5.196   0.000
  0.000   0.000  10.000
Atoms.UnitVectors>
```

# 17 Molecular dynamics

OpenMX Ver. 3.9 supports five molecular dynamics simulations: constant energy molecular dynamics (NVE), constant temperature molecular dynamics by a velocity scaling (NVT_VS), constant temperature molecular dynamics by a velocity scaling to be considered independently for every atoms (NVT_VS2), constant temperature molecular dynamics by the Nose-Hoover method (NVT_NH), and a multi-heat bath molecular dynamics (NVT_VS4).

Each of the molecular dynamics simulations will be explained in the following subsections:

## 17.1 NVE molecular dynamics (NVE)

A constant energy molecular dynamics simulation is performed by the following keyword 'MD.Type':

```
MD.Type           NVE       # NOMD|Opt|NVE|NVT_VS|NVT_VS2|NVT_NH
```

Calculated quantities at every MD step are stored in an output file '*System.Name*.ene', where *System.Name* means 'System.Name'. Although you can find the details in 'iterout.c' in the directory 'source', several quantities are summarized for your convenience as follows:

```
 1:    MD step
 2:    MD time
14:    kinetic energy of nuclear motion, Ukc (Hartree)
15:    DFT total energy, Utot (Hartree)
16:    Utot + Ukc (Hartree)
17:    Fermi energy (Hartree)
```

which means that the first and second columns correspond to MD step and MD time, and so on.

## 17.2 NVT molecular dynamics by a velocity scaling (NVT_VS)

A velocity scaling scheme [30] is supported to perform NVT ensemble molecular dynamics simulations by the following keyword:

```
MD.Type           NVT_VS    # NOMD|Opt|NVE|NVT_VS|NVT_VS2|NVT_NH
```

Then, in this NVT molecular dynamics the temperature for nuclear motion can be controlled by

```
<MD.TempControl
  3
  100   2  1000.0  0.0
  400  10   700.0  0.4
  700  40   500.0  0.7
MD.TempControl>
```

The beginning of the description must be '<MD.TempControl', and the last of the description must be 'MD.TempControl>'. The first number '3' gives the number of the following lines to control the temperature. In this case you can see that there are three lines. Following the number '3', in the consecutive lines the first column means MD steps and the second column gives an interval of MD

steps that the velocity scaling is made. For the example above, a velocity scaling is performed at every two MD steps until 100 MD steps, at every 10 MD steps from 100 to 400 MD steps, and at every 40 MD steps from 400 to 700 MD steps. The third and fourth columns give a given temperature $T_{\rm give}$ (K) and a scaling parameter $\alpha$ in the interval, while the temperature in the interval is given by a linear interpolation. In this velocity scaling, the velocity is scaled by

$$s = \sqrt{\frac{T_{\rm given} + (T_{\rm calc} - T_{\rm given}) * \alpha}{T_{\rm calc}}}$$

$$\mathbf{v}_i' = \mathbf{v}_i \times s$$

where $T_{\rm given}$ and $T_{\rm calc}$ are a given and calculated temperatures, respectively. In 'NVT_VS' the temperature is calculated by using velocities of all the atoms. On the other hand, the *local* temperature is estimated by the velocity of each atom in 'NVT_VS2', and the velocity scaling is performed by the local temperature. After the final MD step given in the specification 'MD.TempControl', the NVT ensemble is switched to a NVE ensemble. Calculated quantities at every MD step are stored in an output file '*System.Name*.ene', where '*System.Name*' means 'System.Name'. Although you can find the details in 'iterout.c', several quantities are summarized for your convenience as follows:

```
 1:    MD step
 2:    MD time
14:    kinetic energy of nuclear motion, Ukc (Hartree)
15:    DFT total energy, Utot (Hartree)
16:    Utot + Ukc (Hartree)
17:    Fermi energy (Hartree)
18:    Given temperature for nuclear motion (K)
19:    Calculated temperature for nuclear motion (K)
22:    Nose-Hoover Hamiltonian (Hartree)
```

which means that the first and second columns correspond to MD step and MD time, and so on. As an example, we show a result for the velocity scaling MD of a glycine molecule in Fig. 12 (a). We see that the temperature in a molecule oscillates around the given temperature. Also for visualization of molecular dynamics, an output file '*System.Name*.md' can be easily animated using free software OpenMX Viewer [152, 151] and XCrySDen [105].

### 17.3 NVT molecular dynamics by the Nose-Hoover method (NVT_NH)

The Nose-Hoover molecular dynamics [31] is supported to perform NVT ensemble molecular dynamics simulations by the following keyword:

```
MD.Type          NVT_NH     # NOMD|Opt|NVE|NVT_VS|NVT_NH
```

Then, in this NVT molecular dynamics the temperature for nuclear motion can be controlled by

```
<MD.TempControl
  4
  1    1000.0
```

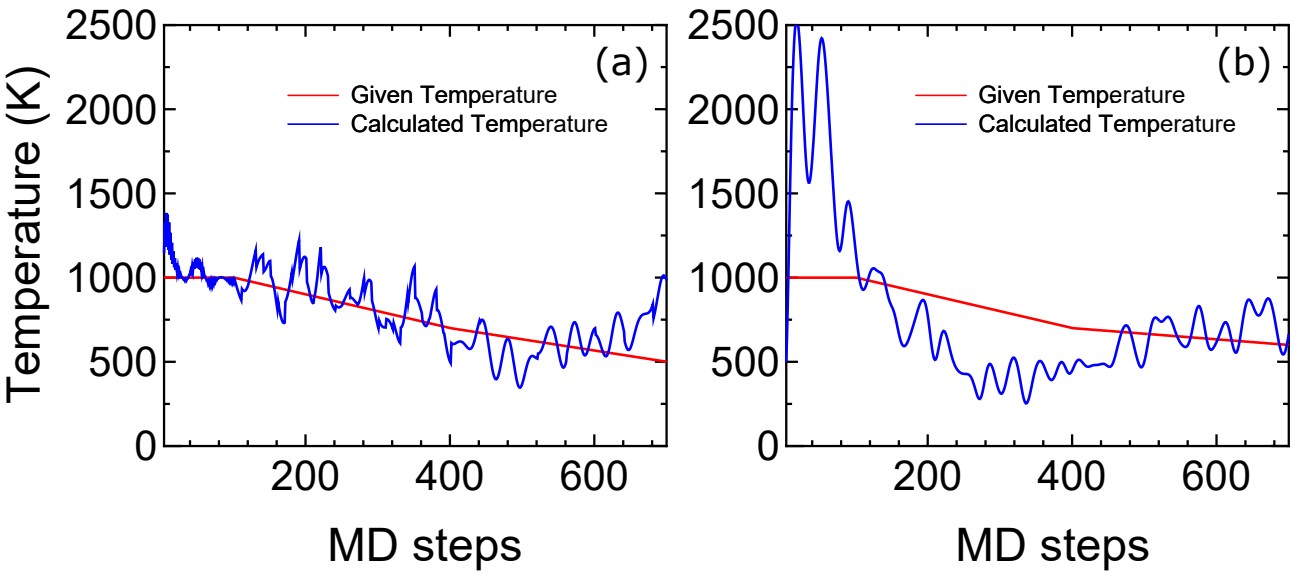



Figure 12: (a) Given and calculated temperatures of a glycine molecule as a function of MD steps in a velocity scaling NVT molecular dynamics. (b) Given and calculated temperatures of a glycine molecule as a function of MD steps in the Nose-Hoover NVT molecular dynamics. The input files are 'Gly_VS.dat' and 'Gly_NH.dat' in the directory 'work', respectively.

```
  100  1000.0
  400   700.0
  700   600.0
MD.TempControl>
```

The beginning of the description must be '<MD.TempControl', and the last of the description must be 'MD.TempControl>'. The first number '4' gives the number of the following lines to control the temperature. In this case you can see that there are four lines. Following the number '4', in the consecutive lines the first and second columns give MD steps and a given temperature for nuclear motion. The temperature between the MD steps is given by a linear interpolation. Although the same keyword 'MD.TempControl' as used in the velocity scaling MD is utilized in this specification, it is noted that the format is different from each other. In addition to the specification of 'MD.TempControl', you must specify a mass of heat bath by the following keyword:

```
NH.Mass.HeatBath          30.0      # default = 20.0
```

The dimension is length$^2$ $\times$ mass. In this specification we use the bohr radius for the length, and the unified atomic mass unit, that the principal isotope of carbon atom is 12.0, for the mass. Calculated quantities at every MD step are stored in an output file '*System.Name*.ene' as explained in 'NVT molecular dynamics by a velocity scaling'. As an example, we show a result for Nose-Hoover MD of a glycine molecule in Fig. 12 (b). We see that the temperature in the molecule oscillates around the given temperature. Also for visualization of molecular dynamics, an output file '*System.Name*.md' can be easily animated using free software such as OpenMX Viewer [152, 151] and XCrySDen [105] as well as NVT_VS.

## 17.4 Multi-heat bath molecular dynamics (NVT_VS)

OpenMX Ver. 3.9 supports a multi-heat bath molecular dynamics simulation where the temperature of each grouped atom is controlled with a heat-bath by a velocity scaling scheme [30]. The method is performed by the following keyword:

```
MD.Type           NVT_VS4
```

The number of groups is specified by

```
MD.num.AtomGroup    2
```

and the groups are defined by

```
<MD.AtomGroup
  1  1
  2  1
  3  1
  4  2
  5  2
MD.AtomGroup>
```

The beginning of the description must be '<MD.AtomGroup', and the last of the description must be 'MD.AtomGroup>'. The first column is a sequential serial number for identifying atoms. The second column is an identification number for each atom, representing the group to which the atom belongs. The identification number has to be specified from 1, and followed by 2, 3, $\cdots$. The above is an example where only five atoms are involved in the system and there are two groups. In Ver. 3.9, the profile of temperature for all the groups is controlled by the keyword 'MD.TempControl' as discussed in the subsection 'NVT molecular dynamics by a velocity scaling'. In the future release, we will support a functionality that temperature is independently controlled for each group.

## 17.5 Constraint molecular dynamics

A constraint scheme is available in the molecular dynamics simulations in which atoms can be fixed in the initial position. The specification is the same as in the subsection 'Constrained relaxation'. See the subsection for the specification.

## 17.6 Initial velocity

For molecular dynamics simulations, it is possible to provide the initial velocity of each atom by the following keyword:

```
<MD.Init.Velocity
 1    3000.000  0.0  0.0
 2   -3000.000  0.0  0.0
MD.Init.Velocity>
```

The example is for a system consisting of two atoms. If you have $N$ atoms, then you have to provide $N$ rows in this specification. The 1st column is the same sequential number to specify atom as in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. The 2nd, 3rd, and 4th columns are $x$-, $y$-, and $z$-components of the velocity of each atom, respectively. The unit of the velocity is m/s. The keyword 'MD.Init.Velocity' is compatible with the keyword 'MD.Fixed.XYZ'.

### 17.7 User definition of atomic mass

In molecular dynamics simulations, OpenMX uses the atomic mass defined in 'Set_Atom_Weight() of SetPara_DFT.c'. However, one can easily change the atomic mass by the keyword 'Definition.of.Atomic.Species'. In such a case, the atomic mass is defined by the fourth column as

```
<Definition.of.Atomic.Species
 H   H5.0-s1          H_PBE19      2.0
 C   C5.0-s1p1        C_PBE19     12.0
Definition.of.Atomic.Species>
```

If the fourth column is not given explicitly, then the default atomic mass will be used. This may be useful to investigate the effect of atomic mass in molecular dynamics, and also may allow us to use a larger time step by using especially the deuterium mass for hydrogen atom. For the definition of atomic mass, we use the unified atomic mass unit that the principal isotope of carbon atom is 12.0.

### 17.8 Converting the file format: md2axsf

In molecular dynamics simulations or geometry optimization, '*System.Name*.md' is generated to save a series of structural change. Although '*System.Name*.md' being the xyz format can be read in [152, 151] and XCrySDen [105], the copied cell in periodic systems cannot be displayed in XCrySDen. For such a purpose, a small post processing code is available to convert the format from 'xyz' to 'axsf'. The first step to do that is to compile 'md2axsf.c' as

```
% gcc md2axsf.c -lm -o md2axsf
```

Then, you can convert a '*System.Name*.md' file as

```
% ./md2axsf System.Name.md System.Name.axsf
```

The '*System.Name*.axsf' file can be analyzed by using XCrySDen and other software.

## 18 Visualization

The electron densities, molecular orbitals, and potentials are output to files in the Gaussian cube format. Figure 13 shows examples of isosurface maps visualized by XCrySDen [105]. These data are output in the Gaussian cube format. So, many software, such as VESTA [103], Molekel [104], and XCrySDen [105], can be used for the visualization. One can find the details of output files in the cube format in the Section 'Output files'.

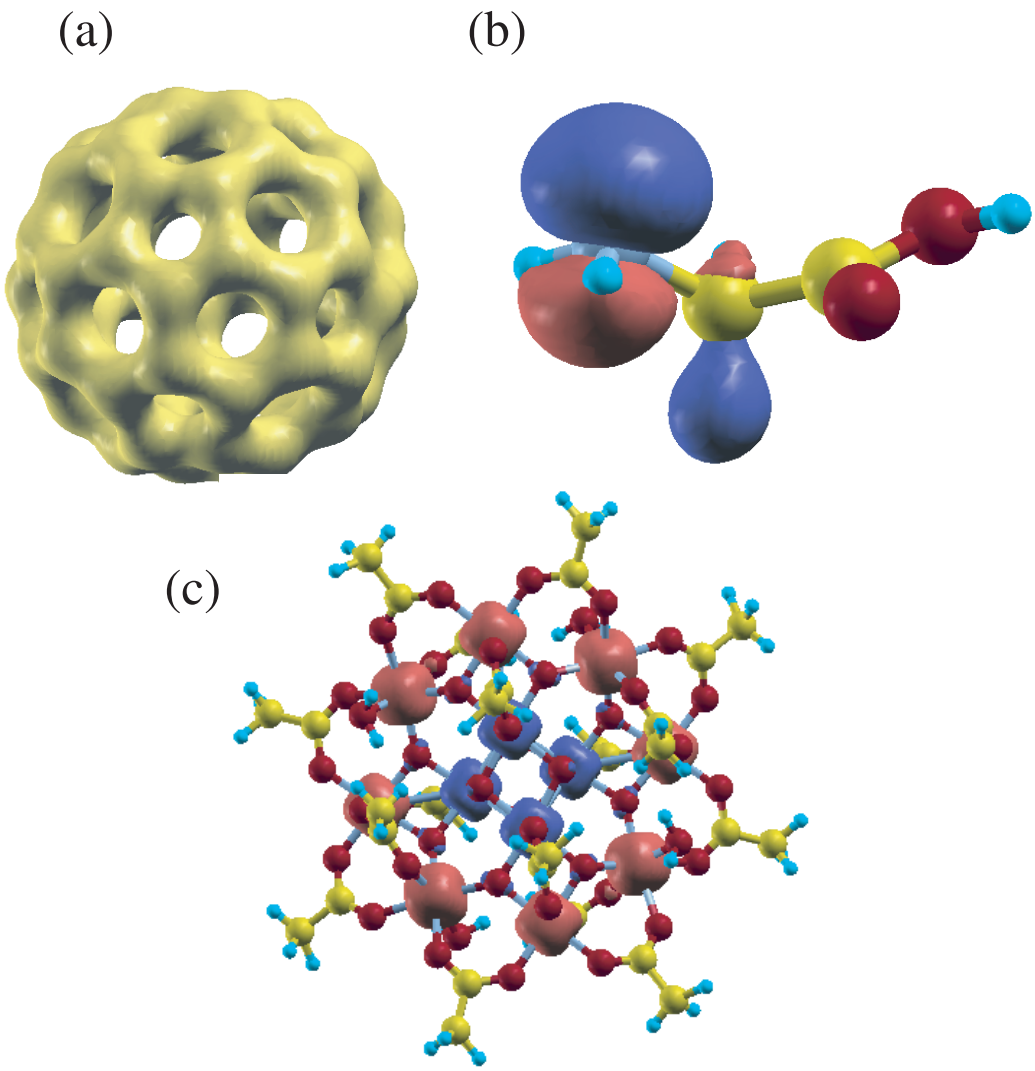



Figure 13: (a) Isosurface map of the total electron density of a $C_{60}$ molecule where 0.13 was used as isovalue of total electron density. (b) Isosurface map of the highest occupied molecular orbital (HOMO) of a glycine molecule where |0.06| was used as isovalue of the molecular orbital. (b) Isosurface map of the spin electron density of a molecular magnet ($Mn_{12}O_{12}(CH_3COO)_{16}(H_2O)_4$ [106]) where |0.02| was used as isovalue of the spin electron density.

# 19 Band dispersion

The band dispersion is calculated by the following two steps:

**(1) SCF calculation**

Let us illustrate the calculation of band dispersion using the carbon diamond. In an input file 'Cdia.dat' store in the directory 'work', the atomic coordinates, cell vectors, and 'scf.Kgrid' are given by

```
Atoms.Number         2
Atoms.SpeciesAndCoordinates.Unit   Ang # Ang|AU
<Atoms.SpeciesAndCoordinates
 1   C  0.000  0.000  0.000   2.0 2.0
 2   C  0.890  0.890  0.890   2.0 2.0
Atoms.SpeciesAndCoordinates>
Atoms.UnitVectors.Unit             Ang # Ang|AU
<Atoms.UnitVectors
  1.7800  1.7800  0.0000
  1.7800  0.0000  1.7800
  0.0000  1.7800  1.7800
Atoms.UnitVectors>

scf.Kgrid                  7 7 7       # means n1 x n2 x n3
```

The unit cell for the band dispersion and **k**-paths are given by

```
Band.dispersion             on        # on|off, default=off
<Band.KPath.UnitCell
 3.56  0.00  0.00
 0.00  3.56  0.00
 0.00  0.00  3.56
Band.KPath.UnitCell>
Band.Nkpath                  5
<Band.kpath
  15  0.0 0.0 0.0   1.0 0.0 0.0   g X
  15  1.0 0.0 0.0   1.0 0.5 0.0   X W
  15  1.0 0.5 0.0   0.5 0.5 0.5   W L
  15  0.5 0.5 0.5   0.0 0.0 0.0   L g
  15  0.0 0.0 0.0   1.0 0.0 0.0   g X
Band.kpath>
```

Then, we execute OpenMX as:

```
% ./openmx Cdia.dat
```

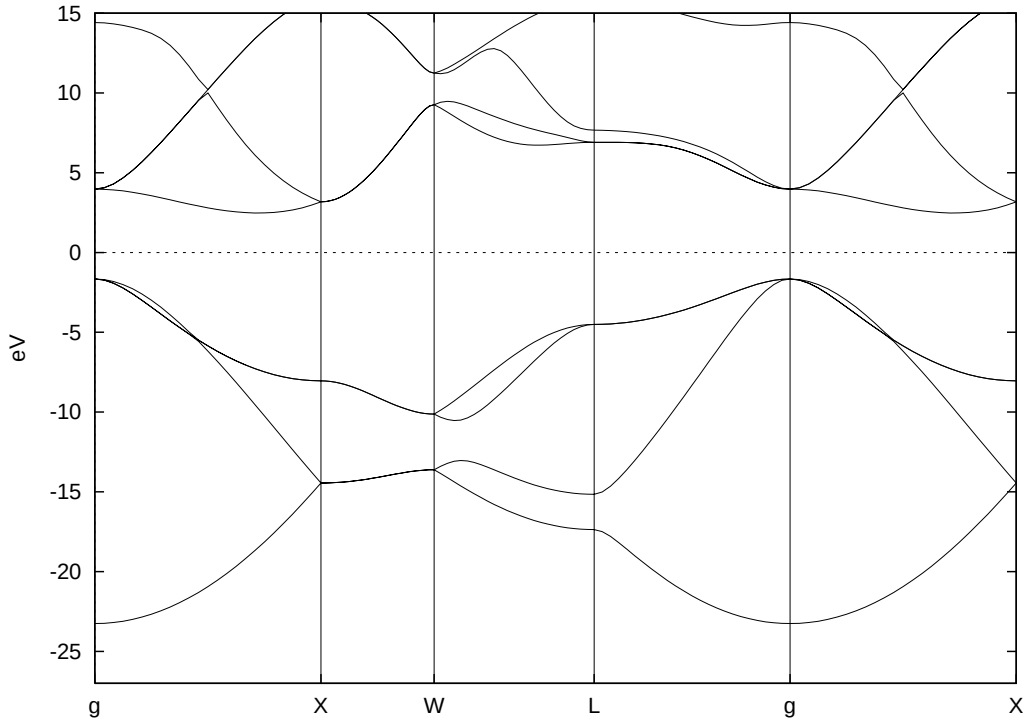



Figure 14: Band dispersion of carbon diamond. The input file is 'Cdia.dat' in the directory 'work'.

When the execution is completed normally, then you can find a file 'cdia.Band' in the directory 'work'. If 'Band.KPath.UnitCell' does not exist, the unit cell specified by the 'Atoms.UnitVectors' will be used.

**(2) Converting of the data to a gnuplot format**

There is a file 'bandgnu13.c' in the directory 'source'. Compile the file as follows:

```
% gcc bandgnu13.c -lm -o bandgnu13
```

When the compile is completed normally, then you can find an executable file 'bandgnu13' in the directory 'source'. Please copy the executable file to the directory 'work'. Using the executable file 'bandgnu13', a file 'cdia.Band' can be converted in a gnuplot format as

```
% ./bandgnu13 cdia.Band
```

Then, two or three files 'cdia.GNUBAND' and 'cdia.BANDDAT1' ('cdia.BANDDAT2'), are generated. The file 'cdia.GNUBAND' is a script for gnuplot, and read the data files 'cdia.BANDDAT1' and 'cdia.BANDDAT2' for the up- and down-spin states, respectively. If spin-polarized calculations using 'LSDA-CA', 'LSDA-PW', or 'GGA-PBE' is employed in the SCF calculation, '*System.Name*.BANDDAT2' for the down-spin state is generated in addition to '*System.Name*.BANDDAT1'. The file 'cdia.GNUBAND' is plotted using gnuplot as follows:

```
% gnuplot cdia.GNUBAND
```

Figure 14 shows the band dispersion of carbon diamond generated by the above procedure, while the range of y-axis was changed in the file 'cdia.GNUBAND'. It is also noted that the chemical potential is automatically shifted to the origin of energy.

A problem in drawing of the band dispersion is how to choose a unit cell used in calculating of the band dispersion. Often, the unit cell used in calculating of the band dispersion is different from that used in the definition of the periodic system. In such a case, you need to define a unit cell used in calculating of the band dispersion by the keyword 'Band.KPath.UnitCell'. If you define 'Band.KPath.UnitCell', the reciprocal lattice vectors for the calculation of the band dispersion are calculated by the unit vectors specified in 'Band.KPath.UnitCell'. If you do not define 'Band.KPath.UnitCell', the reciprocal lattice vectors, which are calculated by the unit vectors specified in 'Atoms.UnitVectors', is employed for the calculation of the band dispersion. In case of fcc, bcc, base centered cubic, and trigonal cells, the reciprocal lattice vectors for the calculation of the band dispersion should be specified using the keyword 'Band.KPath.UnitCell' based on the consuetude in the band structure calculations.

# 20 Density of states

## 20.1 Conventional scheme

The density of states (DOS) is calculated by the following two steps:

**(1) SCF calculation**

Let us illustrate the calculation of DOS using the carbon diamond. In a file 'Cdia.dat' stored in the directory 'work', the keywords for the DOS calculation are set to

```
Dos.fileout                on
Dos.Erange              -25.0  20.0
Dos.Kgrid                12 12 12
```

In the specification of the keyword 'Dos.Erange', the first and second values are the lower and upper bounds of the energy range (eV) for the DOS calculation, respectively, where the origin (0.0) of energy corresponds to the chemical potential. Also, in the specification of the keyword 'Dos.Kgrid', a set of numbers (n1,n2,n3) is the number of grids to discretize the first Brillouin zone in the **k**-space, which is used in the DOS calculation. Then, we execute OpenMX by:

```
% ./openmx Cdia.dat
```

When the execution is completed normally, then you can find files 'cdia.Dos.val' and 'cdia.Dos.vec' in the directory 'work'. The eigenvalues and eigenvectors are stored in the files 'cdia.Dos.val' and 'cdia.Dos.vec' in a text and binary formats, respectively. The DOS calculation is supported even for the O($N$) calculation, while a Gaussian broadening method is employed in this case.

**(2) Calculation of the DOS**

Let us compile a program package for calculating DOS. Move to the directory 'source', and then compile as follows:

```
% make DosMain
```

When the compile is completed normally, then you can find an executable file 'DosMain' in the directory 'source'. Please copy the file 'DosMain' to the directory 'work', and then move to the directory 'work'. You can calculate DOS and projected DOS (PDOS) using the program 'DosMain' from two files 'cdia.Dos.val' and 'cdia.Dos.vec' as:

```
% ./DosMain cdia.Dos.val cdia.Dos.vec
```

Then, you are interactively asked from the program as follow:

```
% ./DosMain cdia.Dos.val cdia.Dos.vec
Max of Spe_Total_CNO = 8
1 1 101 102 103 101 102 103
<cdia.Dos.val>
```

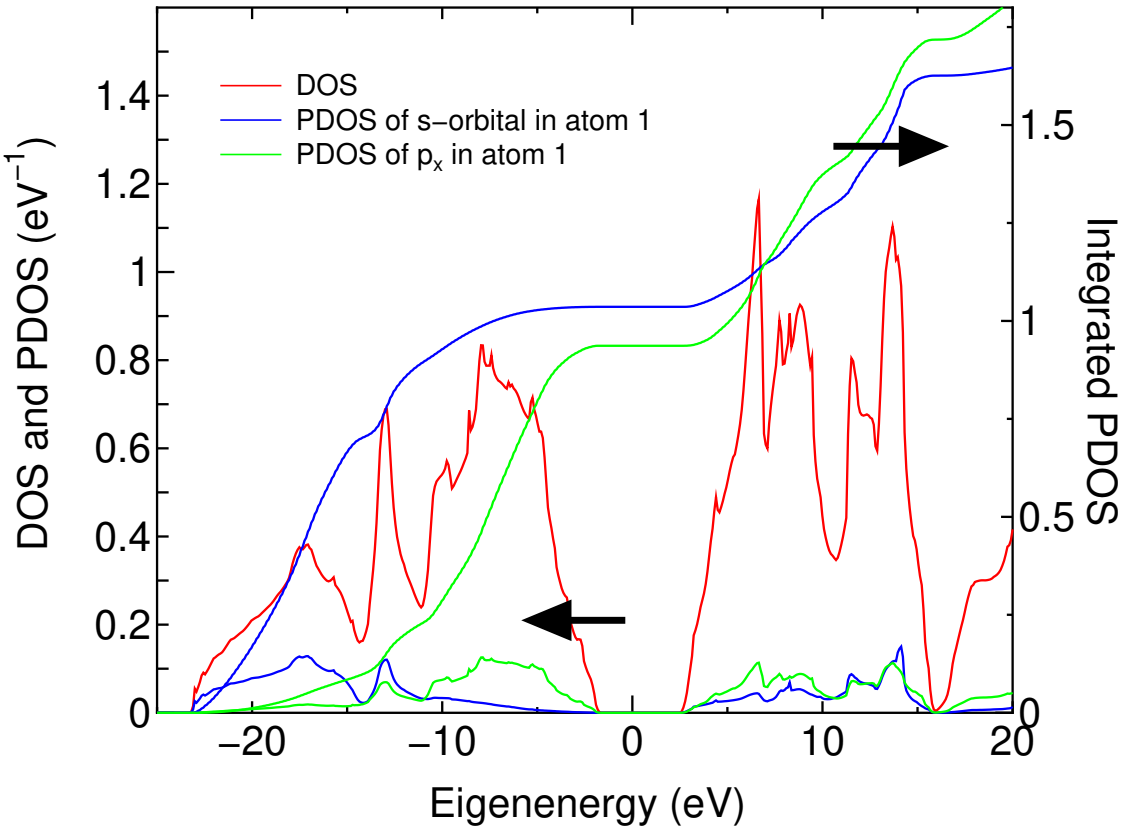



Figure 15: DOS and PDOS of the carbon diamond, and the integrated PDOS, where the Fermi level is set to zero. Since charge redistribution occurs among the s-, p-, and d-orbitals, the integrated PDOS of s- and p-orbitals at the Fermi level are not exactly 1. The calculation can be traced by using an input file 'Cdia.dat' in the directory 'work'.

```
<cdia>
Which method do you use?, Tetrahedron(1), Gaussian Broadeninig(2)
1
Do you want Dos(1) or PDos(2)?
2

Number of atoms=2
Which atoms for PDOS : (1,...,2), ex 1 2
1
pdos_n=1
1
<Spectra_Tetrahedron> start
Spe_Num_Relation 0 0 1
Spe_Num_Relation 0 1 1
Spe_Num_Relation 0 2 101
Spe_Num_Relation 0 3 102
```

```
Spe_Num_Relation 0 4 103
Spe_Num_Relation 0 5 101
Spe_Num_Relation 0 6 102
Spe_Num_Relation 0 7 103
make cdia.PDOS.Tetrahedron.atom1.s1
make cdia.PDOS.Tetrahedron.atom1.p1
make cdia.PDOS.Tetrahedron.atom1.p2
make cdia.PDOS.Tetrahedron.atom1.p3
make cdia.PDOS.Tetrahedron.atom1
```

The tetrahedron [67] and Gaussian broadening methods for evaluating DOS are available. Also, you can select DOS or PDOS. When you select the calculation of PDOS, then please select atoms for evaluating PDOS. In this case, each DOS projected on orbitals ($s$, $px$ ($p1$), $py$ ($p2$), $pz$ ($p3$),..) in selected atoms are output in each file. In these files, the first and second columns are energy in eV and DOS ($\mathrm{eV}^{-1}$) or PDOS ($\mathrm{eV}^{-1}$), and the third column is the integrated DOS or PDOS. If a spin-polarized calculation using 'LSDA-CA', 'LSDA-PW', or 'GGA-PBE' is employed in the SCF calculation, the second and third columns in these files correspond to DOS or PDOS for up and down spin states, respectively, and the fourth and fifth columns are the corresponding integrated values. If you select the Gaussian broadening method, you are requested to set a parameter, $a$ (eV), which determines the width of Gaussian defined by $\exp(-(E/a)^2)$. Figure 15 shows DOS and PDOS of the carbon diamond.

### 20.2 For calculations with lots of k-points

Since the calculation of density of states (DOS) of a large-scale system with lots of k-points requires a considerable memory size, the post-processing code 'DosMain' for generating the partial and total DOS tends to suffer from a segmentation fault. For such a case, a Gaussian DOS scheme is available in which the partial DOS is calculated by the Gaussian broadening method in the OpenMX on-the fly calculation and the information of wave functions is not stored in the file '*System.Name*.Dos.vec'. Since this scheme does not require a large sized memory, it can be used to calculate DOS of large-scale systems. Then, you can specify the following keywords in your input file as

```
DosGauss.fileout      on       # default=off, on|off
DosGauss.Num.Mesh    200       # default=200
DosGauss.Width       0.2       # default=0.2 (eV)
```

When you use the scheme, please specify 'on' for the keyword 'DosGauss.fileout'. The keyword 'DosGauss.Num.Mesh' gives the number of partitioning for the energy range specified by the keyword 'Dos.Erange'. The keyword 'DosGauss.Width' gives a parameter $a$, which is the width of the Gaussian defined by $\exp(-(E/a)^2)$. The keyword 'DosGauss.fileout' and the keyword 'Dos.fileout' are mutually exclusive. Therefore, when you use the scheme, the keyword 'Dos.fileout' must be 'off' as follows:

```
Dos.fileout          off       # on|off, default=off
```

Also, the following two keywords are valid for both the keywords 'Dos.fileout' and 'DosGauss.file'.

```
Dos.Erange        -20.0  20.0   # default=-20 20
Dos.Kgrid            5 5 5      # default=Kgrid1 Kgrid2 Kgrid3
```

It should be noted that the keyword 'DosGauss.fileout' generates only the Gaussian broadening DOS, which means that DOS by the tetrahedron method cannot be calculated by the keyword 'DosGauss.fileout'. After the OpenMX calculation with these keywords, the procedure for 'DosMain' is the same as in the conventional scheme.

# 21 Orbitally decomposed total energy

OpenMX Ver. 3.9 provides a useful tool which decomposes the total energy into each contribution associated with each basis function, where the decomposition is performed based on projection onto basis functions [68]. To calculate the orbitally decomposed total energy, you only have to include the following keyword:

```
Energy.Decomposition      on        # on|off, default=off
```

Let us illustrate the energy decomposition by performing a total energy calculation of a methane molecule. The calculation as an example can be performed as follows:

```
% mpirun -np 5 openmx Methane_ED.dat > met_ed.std &
```

where the input file 'Methane_ED.dat' can be found in the directory 'work'. After finishing the calculation, you may obtain 'met_ed.out'. The part of the file is shown below:

```
*************************************************************************
*************************************************************************
           Decomposed energies in Hartree unit

   Utot = Utot(up) + Utot(dn)
        = Ukin(up) + Ukin(dn) + Uv(up) + Uv(dn)
        + Ucon(up)+ Ucon(dn) + Ucore+UH0 + Uvdw

   Uele = Ukin(up) + Ukin(dn) + Uv(up) + Uv(dn)
   Ucon arizes from a constant potential added in the formalism

          up: up spin state, dn: down spin state
*************************************************************************
*************************************************************************

  Total energy (Hartree) = -8.216132481346387

  Decomposed.energies.(Hartree).with.respect.to.atom

                Utot             Utot(up)    Utot(dn)    Ukin(up)    Ukin(dn)    Uv(up)     ....
    1    C     -6.132295762765   -3.066148   -3.066148    2.076016    2.076016   -2.957459   ....
    2    H     -0.520959186503   -0.260480   -0.260480    0.300675    0.300675   -0.499086   ....
    3    H     -0.520959174111   -0.260480   -0.260480    0.300675    0.300675   -0.499086   ....
    4    H     -0.520959173764   -0.260480   -0.260480    0.300675    0.300675   -0.499086   ....
    5    H     -0.520959184204   -0.260480   -0.260480    0.300675    0.300675   -0.499086   ....


  Decomposed.energies.(Hartree).with.respect.to.atomic.orbital

    1    C          Utot        Utot(up)    Utot(dn)    Ukin(up)    Ukin(dn)    Uv(up)
           multiple
  none             -4.483770   -2.241885   -2.241885    0.000000    0.000000    0.000000      ....
  s           0    -0.675699   -0.337849   -0.337849    0.203145    0.203145   -0.556473      ....
  s           1     0.003690    0.001845    0.001845    0.036240    0.036240   -0.034310      ....
  px          0    -0.325884   -0.162942   -0.162942    0.496144    0.496144   -0.673031      ....
  py          0    -0.325912   -0.162956   -0.162956    0.496166    0.496166   -0.673068      ....
  pz          0    -0.325884   -0.162942   -0.162942    0.496144    0.496144   -0.673031      ....
  px          1     0.005096    0.002548    0.002548    0.107318    0.107318   -0.104552      ....
  ...
  ..
  .
```

It is found that the total energy is exactly decomposed to atomic and orbital contributions. The energy decomposition method will be useful to analyze how local distortion such as impurities and vacancies affects the energy stability/instability for the neighbors. It is also anticipated that the orbital decomposition of the total energy allows us to analyze physical mechanism for a wide variety of phenomena. However, the release of the functionality can be still regarded as experimental one. We may develop and modify the functionality in the near future.

## 22 Orbital optimization

The radial function of basis orbitals can be variationally optimized using the orbital optimization method [41]. As an illustration of the orbital optimization, let us explain it using a methane molecule of which input file is 'Methane_OO.dat'. In the orbital optimization method the optimized orbitals are expressed by a linear combination of primitive orbitals, and obtained by variationally optimizing the contraction coefficients. The number of the primitive and optimized orbitals in the optimization are specified by

```
<Definition.of.Atomic.Species
  H   H5.0-s4>1            H_CA19
  C   C5.0-s4>1p4>1        C_CA19
Definition.of.Atomic.Species>
```

For 'H' one optimized radial function for the $s$-orbital is obtained from a linear combination of four primitive radial functions. Similarly, an optimized radial function for the $s$-($p$-)orbital is obtained from a linear combination of four primitive radial functions for 'C'. In addition, the following keywords are set in the input file as follows:

```
orbitalOpt.Method          species     # Off|Species|Atoms
orbitalOpt.Opt.Method        EF        # DIIS|EF
orbitalOpt.SD.step         0.001       # default=0.001
orbitalOpt.HistoryPulay     30         # default=15
orbitalOpt.StartPulay       10         # default=1
orbitalOpt.scf.maxIter      60         # default=40
orbitalOpt.Opt.maxIter     140         # default=100
orbitalOpt.per.MDIter       20         # default=1000000
orbitalOpt.criterion      1.0e-4       # default=1.0e-4

CntOrb.fileout               on        # on|off, default=off
Num.CntOrb.Atoms             2         # default=1
<Atoms.Cont.Orbitals
 1
 2
Atoms.Cont.Orbitals>
```

Then, we execute OpenMX as:

```
% ./openmx Methane_OO.dat
```

When the execution is completed normally, you can find the history of orbital optimization in the file 'met_oo.out' as:

```
*********************************************************
*********************************************************
         History of orbital optimization   MD= 1
*********     Gradient Norm ((Hartree/borh)^2)     ********
```

```
              Required criterion=  0.000100000000
*********************************************************

   iter=   1  Gradient Norm=  0.057098961101  Uele= -3.217161102876
   iter=   2  Gradient Norm=  0.044668461503  Uele= -3.220120116009
   iter=   3  Gradient Norm=  0.034308306321  Uele= -3.223123238394
   iter=   4  Gradient Norm=  0.025847573248  Uele= -3.226177980300
   iter=   5  Gradient Norm=  0.019106400842  Uele= -3.229294858054
   iter=   6  Gradient Norm=  0.013893824906  Uele= -3.232489198284
   iter=   7  Gradient Norm=  0.010499500005  Uele= -3.235304178159
   iter=   8  Gradient Norm=  0.008362635043  Uele= -3.237652870812
   iter=   9  Gradient Norm=  0.006959703539  Uele= -3.239618540761
   iter=  10  Gradient Norm=  0.005994816379  Uele= -3.241268535418
   iter=  11  Gradient Norm=  0.005298095979  Uele= -3.242657118263
   iter=  12  Gradient Norm=  0.003059655878  Uele= -3.250892948269
   iter=  13  Gradient Norm=  0.001390201488  Uele= -3.255123241210
   iter=  14  Gradient Norm=  0.000780925380  Uele= -3.255179362845
   iter=  15  Gradient Norm=  0.000726631072  Uele= -3.255263012792
   iter=  16  Gradient Norm=  0.000390930576  Uele= -3.250873416989
   iter=  17  Gradient Norm=  0.000280785975  Uele= -3.250333677139
   iter=  18  Gradient Norm=  0.000200668585  Uele= -3.252345643243
   iter=  19  Gradient Norm=  0.000240367596  Uele= -3.254238199726
   iter=  20  Gradient Norm=  0.000081974594  Uele= -3.258146794679
```

In most cases, 20-50 iterative steps are enough to achieve a sufficient convergence. The comparison between the primitive basis orbitals and the optimized orbitals in the total energy is given by

```
Primitive basis orbitals
   Utot  =      -7.992569945749 (Hartree)

Optimized orbitals by the orbital optimization
   Utot  =      -8.133746986502 (Hartree)
```

We see that the small but accurate basis set orbitals can be generated by the orbital optimization. In Fig. 16 we show the convergence properties of total energies for molecules and bulks as a function of the number of unoptimized and optimized orbitals, implying that a remarkable convergent results are obtained using the optimized orbitals for all the systems. In this illustration of a methane molecule, the optimized radial orbitals are output to files 'C_1.pao' and 'H_2.pao'. These output files 'C_1.pao' and 'H_2.pao' could be an input data for pseudo-atomic orbitals as is. This means that it is possible to perform a pre-optimization of basis orbitals for systems you are interested in. The pre-optimization could be performed for smaller but chemically similar systems.

The following two options are available for the keyword 'orbitalOpt.Method': 'atoms' in which basis obitals on each atom are fully optimized, 'species' in which basis obitals on each species are optimized.

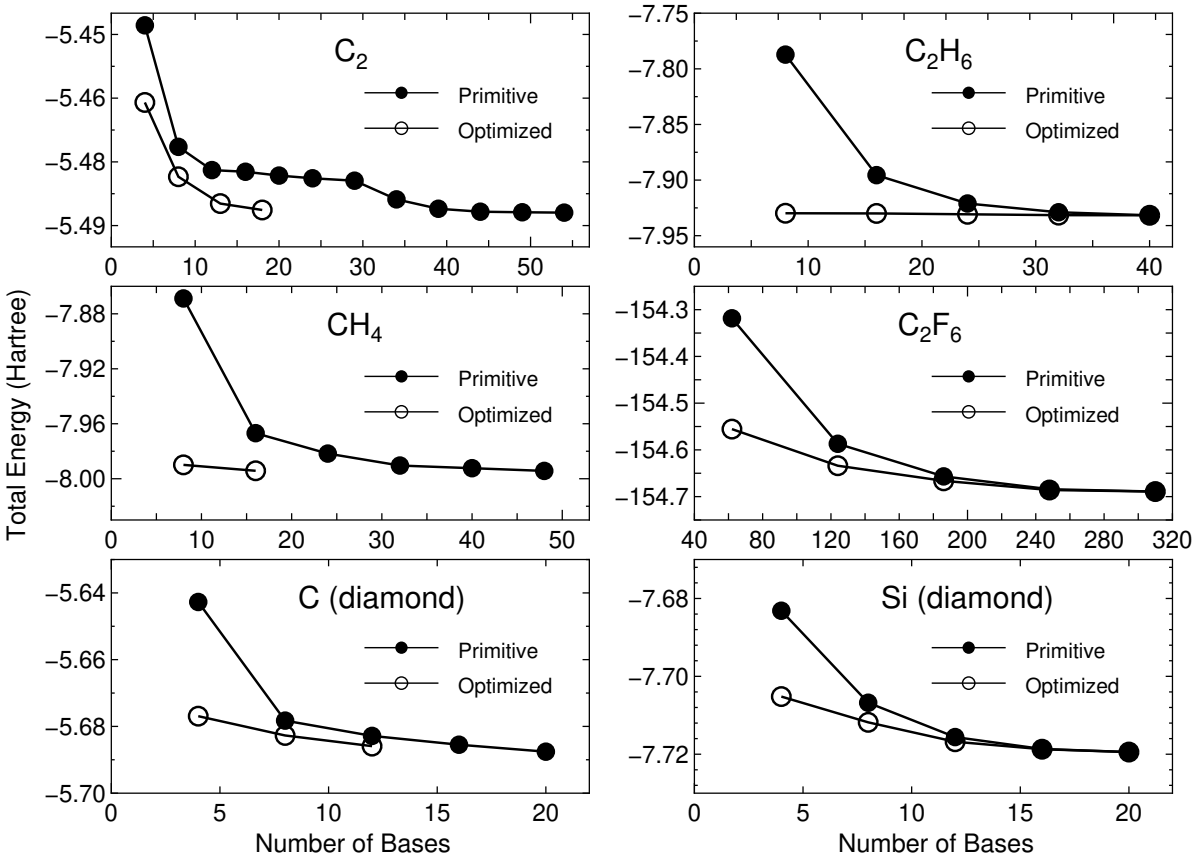



Figure 16: The total energy for a carbon dimer $C_2$, a methane molecule $CH_4$, carbon and silicon in the diamond structure, an ethane molecule $C_2H_6$, and a hexafluoro ethane molecule $C_2F_6$ as a function of the number of primitive and optimized orbitals. The total energy and the number of orbitals are defined as those per atom for $C_2$, carbon and silicon in the diamond, and as those per molecule for $CH_4$, $C_2H_6$, and $C_2F_6$.

- atoms

  The radial functions of basis orbitals are optimized with a constraint that the radial wave function $R$ is independent of the magnetic quantum number, which guarantees the rotational invariance of the total energy. However, the optimized orbital on all the atoms can be different from eath other.

- species

  Basis orbitals in atoms with the same species name, that you define in 'Definition.of.Atomic.Species', are optimized as the same orbitals. If you want to assign the same orbitals to atoms with almost the same chemical environment, and optimize these orbitals, this scheme could be quite convenient. As well as 'atoms', the optimized radial functions are independent of the magnetic quantum number, which guarantees the rotational invariance of the total energy.

Although the same information is available in the section 'Input file', for convenience the details of the other keywords are listed below:

**orbitalOpt.scf.maxIter**
The maximum number of SCF iterations in the orbital optimization is specified by the keyword 'orbitalOpt.scf.maxIter'.

**orbitalOpt.Opt.maxIter**
The maximum number of iterations for the orbital optimization is specified by the keyword 'orbitalOpt.Opt.maxIter'. The iteration loop for the orbital optimization is terminated at the number specified by 'orbitalOpt.Opt.maxIter' even when a convergence criterion is not satisfied.

**orbitalOpt.Opt.Method**
Two schemes for the optimization of orbitals are available: 'EF' which is an eigenvector following method, 'DIIS' which is the direct inversion method in an iterative subspace. The algorithms are basically same as for the geometry optimization. Either 'EF' or 'DIIS' is chosen by the keyword, 'orbitalOpt.Opt.Method'.

**orbitalOpt.StartPulay**
The quasi Newton methods, 'EF' and 'DIIS' start from the optimization step specified by the keyword 'orbitalOpt.StartPulay'.

**orbitalOpt.HistoryPulay**
The keyword 'orbitalOpt.HistoryPulay' specifies the number of previous steps to estimate the next input contraction coefficients used in the quasi Newton methods 'EF' and 'DIIS'.

**orbitalOpt.SD.step**
Steps before moving to the quasi Newton method 'EF' or 'DIIS' is performed by the steepest decent method. The prefactor used in the steepest decent method is specified by the keyword 'orbitalOpt.SD.step'. In most cases, orbitalOpt.SD.step of 0.001 can be a good prefactor.

**orbitalOpt.criterion**
The keyword 'orbitalOpt.criterion' specifies a convergence criterion ((Hartree/borh)$^2$) for the orbital optimization. The iterations loop is finished when a condition, Norm of derivatives<orbitalOpt.criterion, is satisfied.

**CntOrb.fileout**
If you want to output the optimized radial orbitals to files, then the keyword 'CntOrb.fileout' must be ON.

**Num.CntOrb.Atoms**
The keyword 'Num.CntOrb.Atoms' gives the number of atoms whose optimized radial orbitals are output to files.

**Atoms.Cont.Orbitals**
The keyword 'Atoms.Cont.Orbitals' specifies the atom number, which is given by the first column in the specification of the keyword 'Atoms.SpeciesAndCoordinates' for the output of optimized orbitals as follows:

```
<Atoms.Cont.Orbitals
 1
 2
```

```
Atoms.Cont.Orbitals>
```

The beginning of the description must be '<Atoms.Cont.Orbitals', and the last of the description must be 'Atoms.Cont.Orbitals>'. The number of lines should be consistent with the number specified in the keyword 'Atoms.Cont.Orbitals'. For example, the name of files are 'C_1.pao' and 'H_2.pao', where the symbol corresponds to that given by the first column in the specification of the keyword 'Definition.of.Atomic.Species' and the number after the symbol means that of the first column in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. These output files 'C_1.pao' and 'H_2.pao' can be an input data for pseudo-atomic orbitals as is.

# 23 Order($N$) method

The computational effort of the conventional diagonalization scheme scales as the third power of the number of basis orbitals, which means that the part could be a bottleneck when large-scale systems are calculated. On the other hand, the O($N$) methods can solve the eigenvalue problem in O($N$) operation in exchange for accuracy. Thus, O($N$) methods could be efficient for large-scale systems, while a careful consideration is always required for the accuracy. In OpenMX Ver. 3.9, three O($N$) methods are available: a divide-conquer (DC) method [50], a divide-conquer (DC) method with localized natural orbitals (LNO) [51], and a Krylov subspace method [43]. Our recommendation among the three O($N$) methods is the DC-LNO method, since the method is robust, and can be applied to a wide range of materials including metals, and the parallel efficiency is expected to be the best one among the three methods. In the following subsections each O($N$) method is illustrated by examples.

## 23.1 Divide-conquer method

The DC method is a robust scheme and can be applicable to a wide variety of materials with a reasonable degree of accuracy and efficiency, while this scheme is suitable especially for covalent systems. In this subsection, the O($N$) calculation using the DC method is illustrated. In an input file 'DIA8_DC.dat' which can be found in the directory 'work', please specify 'DC' for the keyword 'scf.EigenvalueSolver'.

```
scf.EigenvalueSolver   DC
```

Then, one can execute OpenMX by:

```
% ./openmx DIA8_DC.dat
```

The input file is for an O($N$) calculation (1 MD step) of the diamond including 8 carbon atoms. The computational time is 120 seconds using a Xeon machine (2.6 GHz). Figure 17 shows the computational time and memory size to calculate a MD step of the carbon diamond as a function of number of atoms in the supercell. In fact, we see that the computational time and memory size are almost proportional to the number of atoms. The accuracy and efficiency of the DC method are controlled by a single parameter: 'orderN.HoppingRanges'.

- orderN.HoppingRanges

  The keyword 'orderN.HoppingRanges' defines the radius of a sphere which is centered on each atom. The physically truncated cluster for each atom is constructed by picking up atoms inside the sphere with the radius in the DC, DC-LNO, and O($N$) Krylov subspace methods.

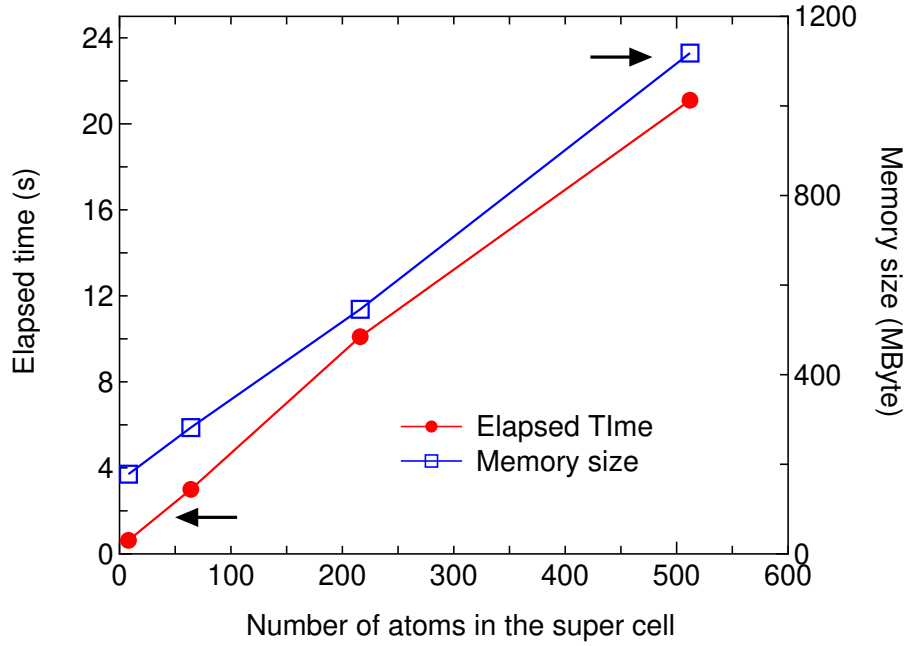



Figure 17: Elapsed time of the diagonalization part per SCF step and computational memory size per MPI process as a function of carbon atoms in the diamond supercell, where 16 processes were used in the MPI parallel calculations. C5.0-s1p1 was used as basis functions. For the DC method, orderN.HoppingRanges=6.0 (Å) is used. A Xeon machine (2.6 GHz) was used to measure the elapsed time. The input files are 'DIA8_DC.dat', 'DIA64_DC.dat', 'DIA216_DC.dat', and 'DIA512_DC.dat' in the directory 'work'.

Table 4: Total energy and computational time per MD step of a $C_{60}$ molecule and small peptide molecules (valorphin [107]) and DNA consisting of cytosines and guanines calculated by the conventional diagonalization and the O($N$) DC method, where a minimal basis set was used. In this Table, numbers in the parenthesis after DC means 'orderN.HoppingRanges' used in the DC calculation. The computational times were measured using an Opteron PC cluster (48 cpus × 2.4 GHz). The input files are 'C60_DC.dat', 'Valorphin_DC.dat', 'CG15c_DC.dat' in the directory 'work'.

| | Total energy (Hartree) | Computational time (s) |
|---|---|---|
| **$C_{60}$** | | |
| (60 atoms, 240 orbitals) | | |
| Conventional | -343.89680 | 36 |
| DC (7.0) | -343.89555 | 37 |
| **Valorphin** | | |
| (125 atoms, 317 orbitals) | | |
| Conventional | -555.28953 | 81 |
| DC (6.5) | -555.29019 | 76 |
| **DNA** | | |
| (650 atoms, 1880 orbitals) | | |
| Conventional | -4090.95463 | 576 |
| DC (6.3) | -4090.95092 | 415 |

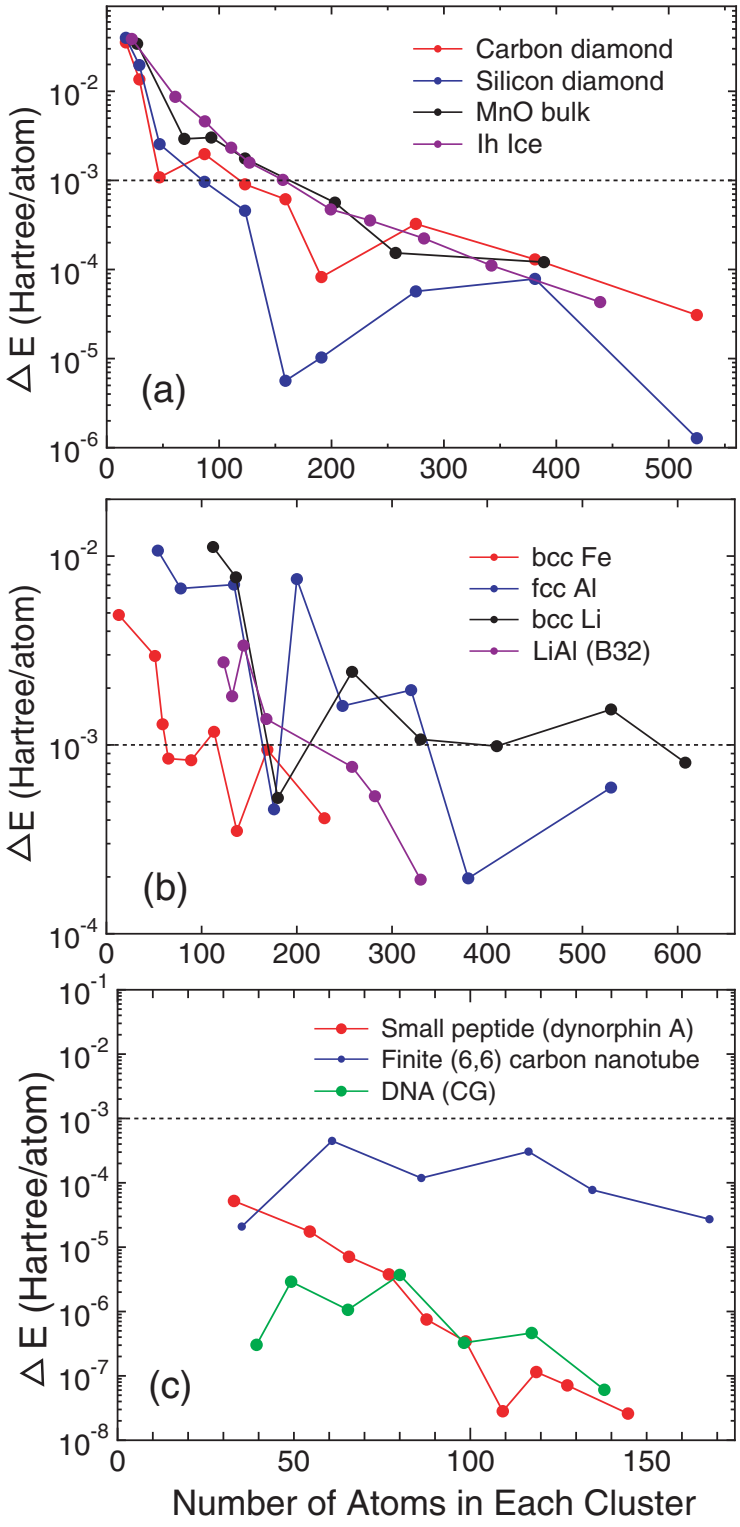



Figure 18: Error in the total energy of (a) bulks with a finite gap, (b) metals, and (c) molecular systems calculated by the divide-conquer (DC) method as a function of the number of atoms in each cluster. The dotted horizontal line indicates 'milli-Hartree' accuracy.

If the number of atoms in the systems is $N$, $N$ small eigenvalue problems for the $N$ physically truncated clusters are solved, and then the total density of states (DOS) is constructed as the sum of the projected DOS of each physically truncated cluster. Although the appropriate value for 'orderN.HoppingRanges' depends on systems, for molecular systems the following values are recommended as a trade-off between the computational accuracy and efficiency:

```
orderN.HoppingRanges     6.0 - 7.0
```

Table 4 shows the comparison in the total energy between the exact diagonalization and the DC method for a $C_{60}$ molecule and small peptide molecules (valorphin [107]), and DNA consisting of cytosines and guanines. We find that errors in the total energy calculated by the DC method are about a few mHartree in the system size. Also, it can be estimated that the DC method is faster than the conventional diagonalization when the number of atoms is larger than 500 atoms, while the crossing point between the conventional diagonalization and the DC method with respect to computational time depends on systems and the number of processors in the parallel calculation.

To see an overall tendency in the convergence properties of total energy with respect to the size of truncated cluster, the error in the total energy, compared to the exact diagonalization, is shown as a function of the number of atoms in each cluster for (a) bulks with a finite gap, (b) metals, and (c) molecular systems in Fig. 18. We see that the error decreases almost exponentially for the bulks with a finite gap and molecular systems, while the convergence speed is slower for metals.

### 23.2 Divide-conquer method with localized natural orbitals (DC-LNO) method

The DC-LNO method [51] is a variant of the DC method. The computational efficiency is improved by introducing localized natural orbitals (LNOs) to span the subspace of each atom. The dimension of the resultant subspace is smaller than that in the DC method, leading to the reduction of computational cost. The LNOs are non-iteratively calculated by a low-rank approximation via a local eigendecomposition of a projection operator for the occupied space. As shown in Fig. 19(a), introducing LNOs to represent the long range region of a truncated cluster reduces the computational cost of the DC method while keeping computational accuracy. The method can be applied to not only gapped systems, but also metallic systems as long as the size of truncated clusters is large enough, and typically clusters including more than 200 atoms might be chosen. The functionality is compatible with not only the collinear calculation, but also the non-collinear calculations.

As a first step for O($N$) calculations of the DC-LNO method, one can perform an O($N$) calculation for Si crystal using an input file store in the directory 'work' as

```
% mpirun -np 112 ./openmx Si8-LNO.dat | tee si8-lno.std
```

The calculation was performed in 66 seconds using 112 cores on a Xeon cluster machine of 2.6 GHz. In order for you to start an trial calculation by the DC-LNO method, an input file 'Si8-LNO.dat' is available in the directory 'work'. Since as shown in Fig. 19(b) the three level parallelization has been implemented: atom level, spin level, and diagonalization level, it is expected that the method scales up to, e.g., 40000 CPU cores for a 1000 atom system in the parallel calculations, where we assumed 1000 atoms $\times$ 2 (spin-polarized calculation) $\times$ 20 CPU cores per node, resulting in that the product becomes 40000. The benchmark calculation of the multi-level parallelization will be shown later on. When you try to perform the hybrid parallelization, the following keyword has to be switched on:

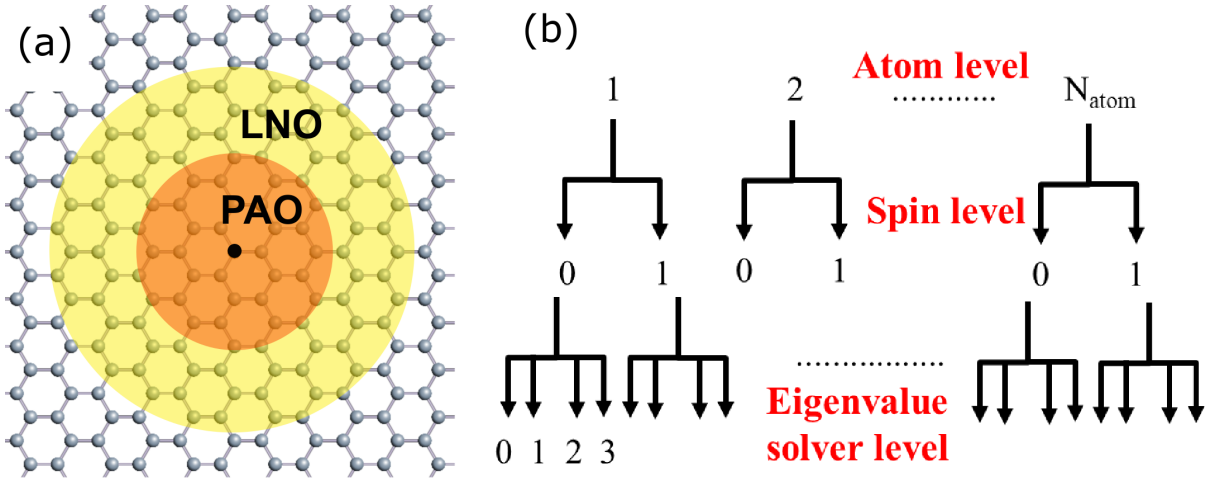



Figure 19: Truncation of a system in the DC method with LNOs. The short range (orange) and long range (yellow) regions are represented by PAOs and LNOs, respectively.

```
scf.dclno.threading       on    # off|on
```

In the hybrid parallelization the diagonalization at the bottom level will be parallelized by OpenMP. The computational accuracy and efficiency of the method can be controlled by the following keywords:

```
orderN.HoppingRanges      7.0   # 7.0 (Ang.)
orderN.LNO.Buffer         0.2   # default = 0.2
orderN.LNO.Occ.Cutoff     0.1   # default = 0.1
```

The role of the keyword 'orderN.HoppingRanges' is exactly the same as that in the DC method. For each atom a truncated cluster is constructed by picking-up atoms within a sphere whose radius is specified by the keyword 'orderN.HoppingRanges'. Though the proper choice of the parameter depends on systems, a serie of benchmark calculations implies that the accuracy is enough for not only gapped systems, but also metallic systems if 'orderN.HoppingRanges' is set so that the resultant truncated cluste can include 300 atoms. The setting might be regarded as a conservative choice to ensure the accuracy rather than efficiency. So, a compromising choice with respect to both accuracy and efficiency may be in between 200 and 300 atoms. The region in the truncated cluster where the PAOs are replaced by LNOs is determined by the keyword 'orderN.LNO.Buffer'. The 'orderN.LNO.Buffer=0.0' means that PAOs allocated on all the SNAN atoms are replaced by LNOs, while the PAOs on all the SNAN atoms remain unchanged in the case with the 'orderN.LNO.Buffer=1.0' which is equivalent to the DC method. As for the SNAN, please refer the subsection 23.4 'User definition of FNAN+SNAN'. The orderN.LNO.Buffer of 0.1∼0.2 might be a proper choice with respect to accuracy and efficiency, while the default value is 0.2. The selection of LNOs for each atom $i$ is performed by monitoring eigenvalues of $\Lambda_{\mathbf{0}i}$ defined with [51]

$$\Lambda_{\mathbf{0}i} = \sum_{\mathbf{R}j} \rho_{\mathbf{0}i,\mathbf{R}j} S_{\mathbf{R}j,\mathbf{0}i}, \tag{2}$$

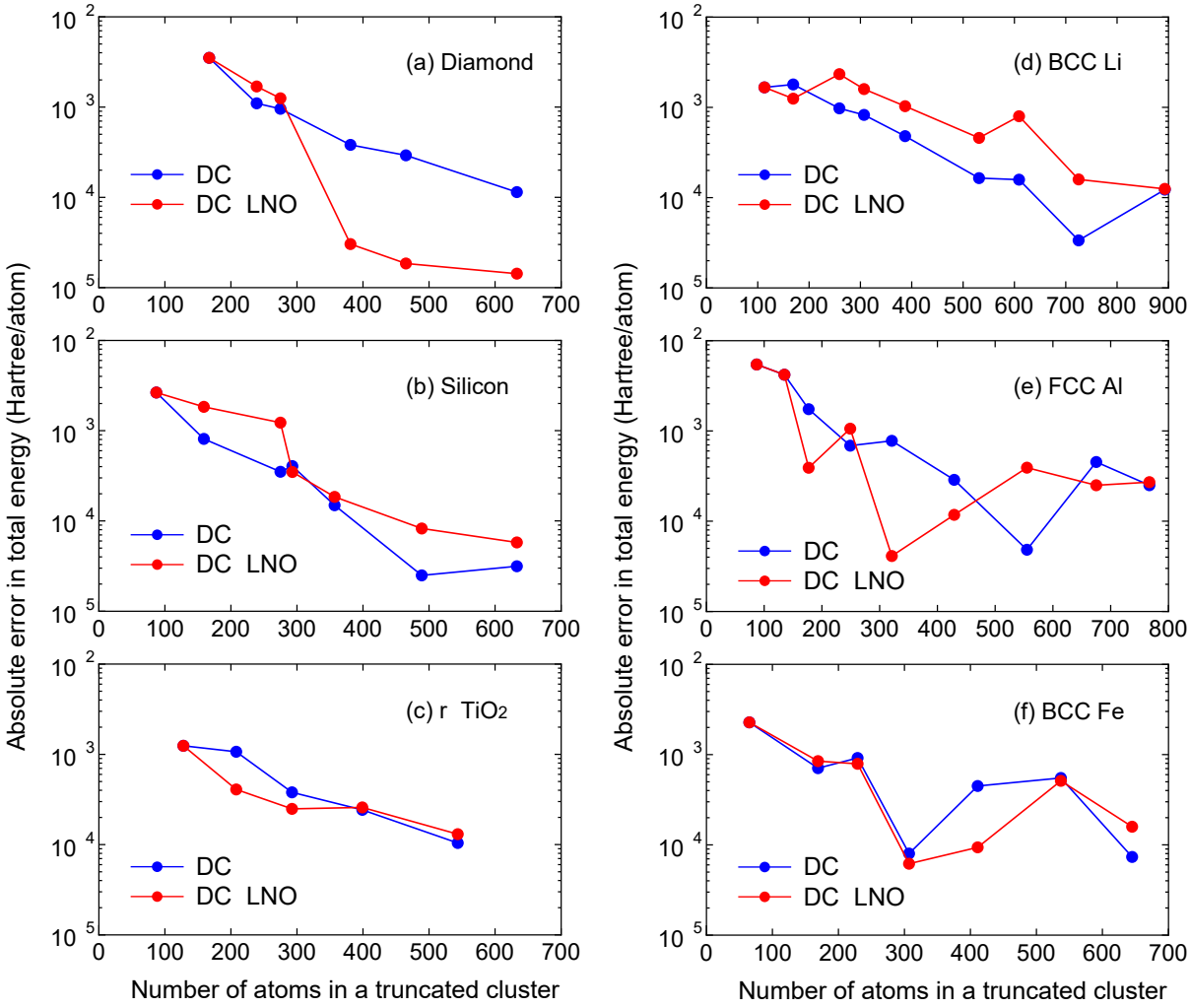



Figure 20: Absolute error in the total energy (Hatree/atom) for (a) diamond, (b) silicon in the diamond structure, (c) rutile TiO2, (d) BCC lithium, (e) FCC aluminum, and (f) BCC iron as a function of the number of atoms in a truncated cluster calculated by the DC and DC-LNO methods. The experimental lattice constants were used for all the cases.

where $\rho_{\mathbf{0}i,\mathbf{R}j}$ and $S_{\mathbf{R}j,\mathbf{0}i}$ are block elements of density matrix and overlap matrix, respectively. Since the eigenvalues can be understood as population of LNOs,. the LNOs having the population more than orderN.LNO.Occ.Cutoff are chosen as basis functions for the targeted atom, which well span the occupied subspace space. Instead of using 'orderN.LNO.Occ.Cutoff', one can directly specify the number of LNOs for each species by the keyword 'LNOs.Num'. When you define the species as

```
<Definition.of.Atomic.Species
  Si     Si7.0-s2p2d1    Si_PBE19
  H      H6.0-s2p1       H_PBE19
Definition.of.Atomic.Species>
```

the number of LNOs can be specified by

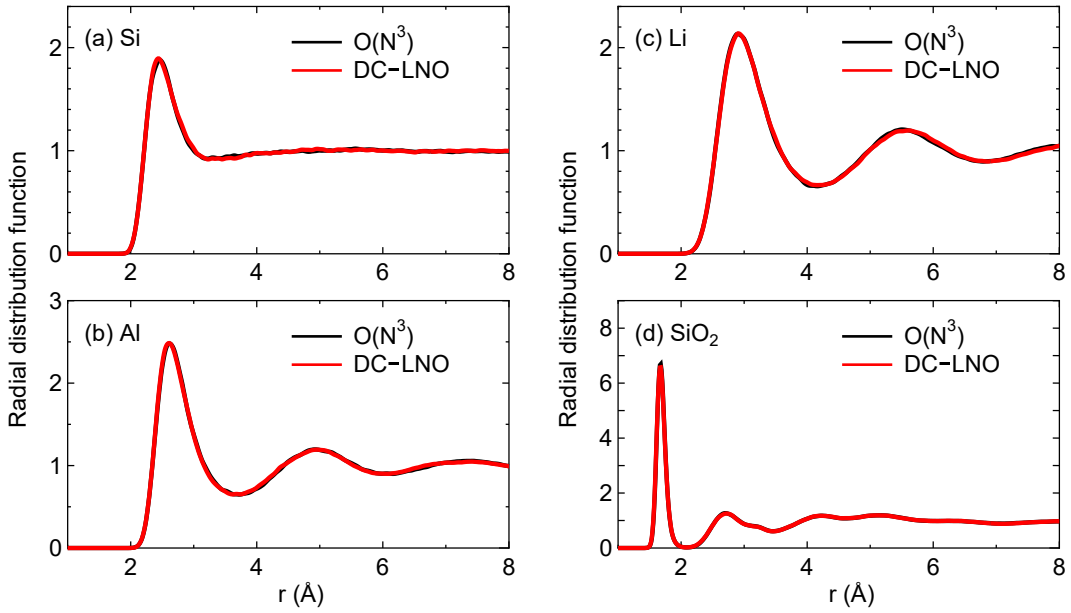



Figure 21: Total radial distribution function (RDF) of (a) silicon at 3500 K, (b) aluminum at 2500 K, (c) lithium at 800 K, and (d) $SiO_2$ at 3000 K, calculated by the conventional O(N3 ) diagonalization and the DC-LNO methods. The MD simulations were performed for cubic supercells containing 64, 108, 128, and 192 atoms with a fixed lattice constant of 10.86, 12.15, 14.04, 14.25 for silicon, aluminum, lithium, and SiO2, respectively, for 10 ps with the time step of 2 fs. The details of the simulations can be in Ref. [51].

```
<LNOs.Num
  Si     4
  H      1
LNOs.Num>
```

In this case, the numbers of the LNOs are fixed to 4 and 1 for Si and H, respectively. To avoid a sudden change of the number of LNOs during geometry optimization and molecular dynamics simulations, it might be better to use 'LNOs.Num' rather than orderN.LNO.Occ.Cutoff. The comparison between the DC and DC-LNO methods is shown in Fig. 20. Although the PAOs in the long range region are replaced by the LNOs, it is found that the accuracy is comparable to the DC method both in gapped and metallic systems. As an illustration for applications of the DC-LNO method, we show in Fig. 21 radial distribution functions (RDF) of liquids for silicon, aluminum, lithium, and $SiO_2$. It turns out that in all the cases the DC-LNO method reproduces well the results by the conventional $O(N^3)$ diagonalization method, and that the obtained RDFs are well compared to other computational results [52, 53, 54, 55].

In Fig. 22 the speed-up ratio in the MPI parallelization of the DC-LNO method is shown for non-spin polarized calculations of a diamond supercell containing 64 atoms. Since the multiplicity of spin index is 1, we see a nearly ideal behavior up to 64 MPI processes. Beyond 64 MPI processes the parallelization in the diagonalization level is taken into account on top of the parallelization in the atom level. A superlinear speed-up is observed at 128 and 256 MPI processes, which might be due

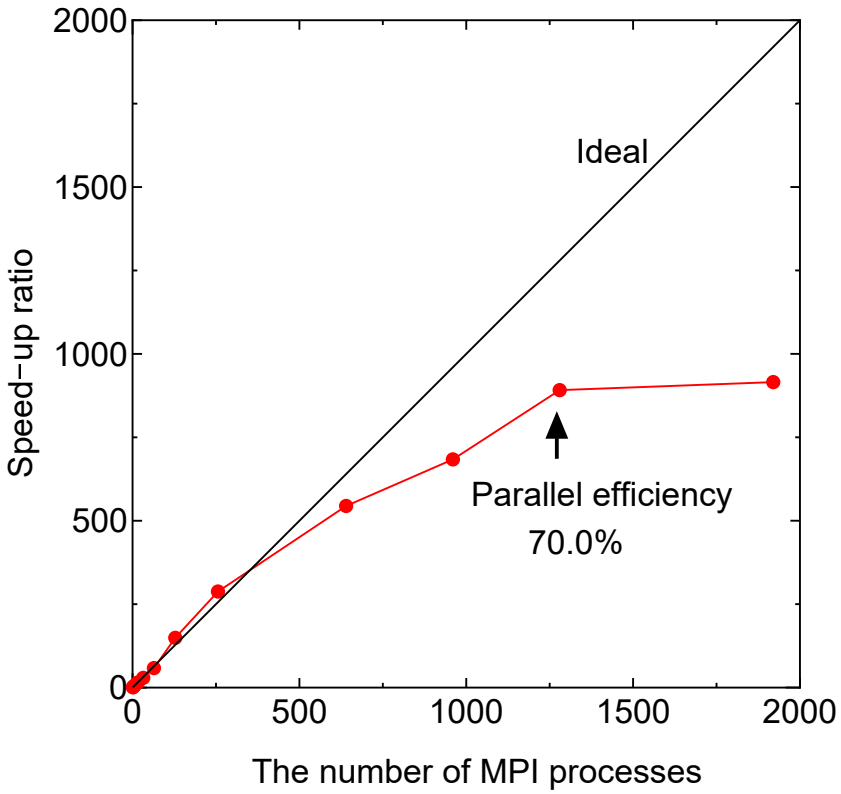



Figure 22: Speed-up ratio in the MPI parallelization of the DC-LNO method for a diamond supercell containing 64 atoms, where the cutoff radius rL of 8.0 Åwas used, leading to the numbers of atoms of 239 and 142 in the short and long range regions, and the dimension of matrices of 3675 for the truncated cluster problem. The details of the benchmark calculation can be in Ref. [51].

to an effective use of cache by the reduction of memory usage, and a good scaling is achieved up to 1280 MPI processes at which the parallel efficiency is calculated to be 70% using the elapsed time at 1 MPI process as reference. Since each computer node has 20 CPU cores in this case, it would be reasonable to observe the good caling up to 1280 (=64×20) MPI processes. Thus, we see that the multilevel parallelization is very effective to minimize the computational time in accordance with a recent development of massively parallel computers.

### 23.3 Krylov subspace method

The DC method is robust and accurate for a wide variety of systems. However, the size of truncated clusters to obtain an accurate result tends to be large for metallic systems as shown in Fig. 18. A way of reducing the computational efforts is to map the original vector space defined by the truncated cluster into a Krylov subspace of which dimension is smaller than that of the original space [43]. The Krylov subspace method is available by

```
scf.EigenvalueSolver        Krylov
```

Basically, the accuracy and efficiency are controlled by the following two keywords:

```
orderN.HoppingRanges          6.0
```

```
orderN.KrylovH.order          400
```

The keyword 'orderN.HoppingRanges' defines the radius of a sphere centered on each atom in the same sense as that in the DC method. The dimension of the Krylov subspace of Hamiltonian in each truncated cluster is given by 'orderN.KrylovH.order'. Moreover, the Krylov subspace method can be precisely tuned by the following keywords:

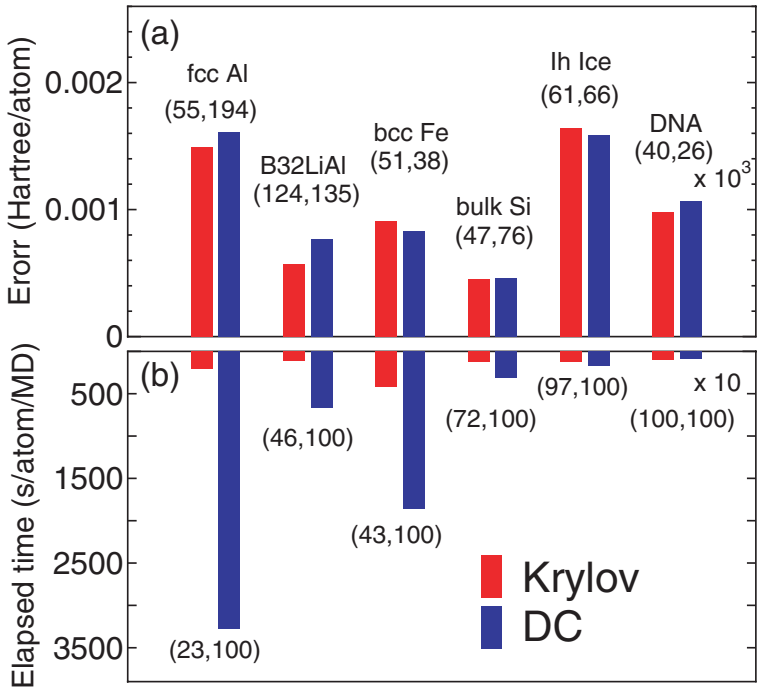



Figure 23: (a) absolute error, with respect to the band calculations, in the total energy (Hartree/atom) calculated by the Krylov subspace and DC methods for metals and finite gap systems, (b) computational time (s/atom/MD). For a substantial comparison, the calculations were performed using a single Xeon processor. The set of numbers in the parenthesis of (a) means the average number of atoms in the core and buffer regions. The set of numbers in the parenthesis of (b) means the percentage of the dimension of the subspaces relative to the total number of basis functions in the truncated cluster, respectively.

- orderN.Exact.Inverse.S  on| off, default=on

  In case of 'orderN.Exact.Inverse.S=on', the inverse of overlap matrix for each truncated cluster is exactly evaluated. Otherwise, see the next keyword 'orderN.KrylovS.order'.

- orderN.KrylovS.order  1200, default=orderN.KrylovH.order×4

  In case of 'orderN.Exact.Inverse.S=off', the inverse is approximated by a Krylov subspace method for the inverse, where the dimension of the Krylov subspace of overlap matrix in each truncated cluster is given by the keyword 'orderN.KrylovS.order'.

- orderN.Recalc.Buffer  on| off, default=on

In case of 'orderN.Recalc.Buffer=on', the buffer matrix is recalculated at every SCF step. Otherwise, the buffer matrix is calculated at the first SCF step, and fixed at the subsequent SCF steps.

- orderN.Expand.Core on| off, default=on

  In case of 'orderN.Expand.Core=on', the core region is defined by atoms within a sphere with radius of $1.2 \times r_{\rm min}$, where $r_{\rm min}$ is the distance between the central atom and the nearest atom. The core region defines a set of vectors used for the first step in the generation of the Krylov subspace for each truncated cluster. In case of 'orderN.Expand.Core=off', the central atom is considered as the core region. The default is 'on'.

It is better to switch on 'orderN.Exact.Inverse.S' and 'orderN.Expand.Core' as the covalency increases, while the opposite could becomes better in simple metallic systems. In Fig. 23 the absolute error in the total energy calculated by the Krylov and DC methods are shown for a wide variety of materials. It is found that in comparison with the DC method, the Krylov subspace method is more efficient especially for metallic systems, and that the efficiency become comparable as the covalency and ionicity in the electronic structure increase.

It is also noted that the O($N$) Krylov subspace method is well parallelized to realize large-scale calculations. The most efficient parallelization for the O($N$) Krylov subspace method can be realized by using the same number of MPI processes as that of atoms together with OpenMP threads. Figure 24 shows that a system consisting of a hundred thousand atoms can be treated on a massively parallel computer [44, 45], where the diamond structure consisting of 131072 carbon atoms is considered as a benchmark system.

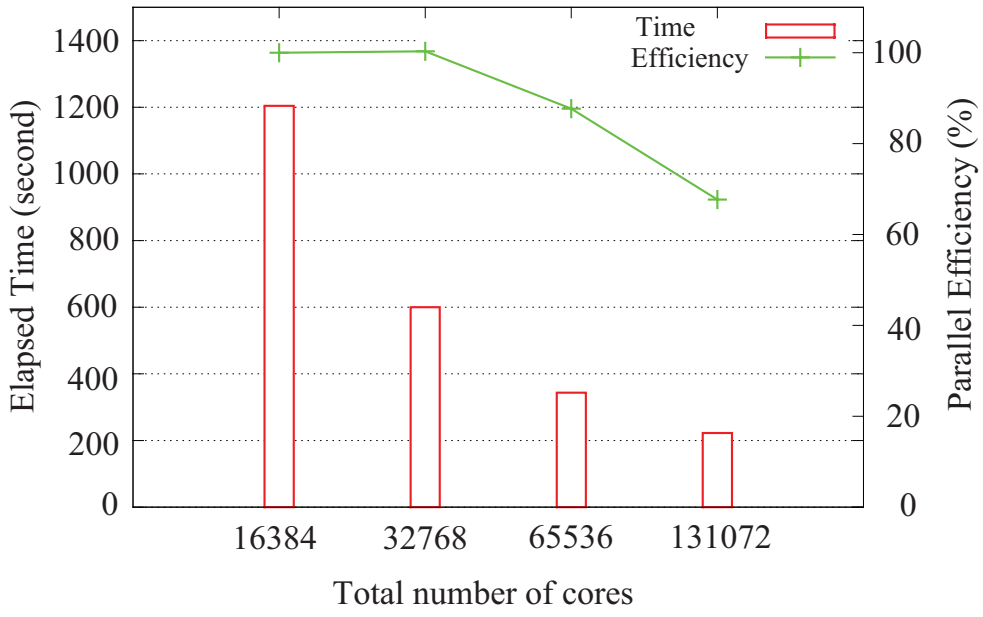



Figure 24: Parallel efficiency of the O($N$) Krylov subspace method in the hybrid parallelization on the K-computer, where eight threads were used for all the cases. The diamond structure consisting of 131072 carbon atoms was considered as a benchmark system.

### 23.4 User definition of FNAN+SNAN

In all the O($N$) methods supported by OpenMX Ver. 3.9, neighboring atoms in each truncated cluster are classified into two categories: *first* and *second* neighboring atoms. If the sum, $r_0 + r_{\rm N}$, of a cutoff radius, $r_0$, of basis functions allocated to the central atom and that, $r_{\rm N}$, of a neighboring atom is smaller than the distance between the two atoms, then the neighboring atom is regarded as a first neighboring atom, and the other atoms, which does not satisfy the criterion, in the truncated cluster are called the second neighboring atom. The second neighboring atoms are determined by a keyword 'orderN.HoppingRanges'. The numbers of the first and second neighboring atoms determined by the keyword are shown in the standard output as *FNAN* and *SNAN*, respectively. In addition to the use of the keyword 'orderN.HoppingRanges' for determining FNAN and SNAN, one can directory control the number, FNAN+SNAN, by the following keyword:

```
<orderN.FNAN+SNAN
  1  60
  2  65
  3  60
  4  50
  ..
  .
orderN.FNAN+SNAN>
```

In this specification, the number of row should be equivalent to that of atoms. The first column is a serial number corresponding to the serial number defined in the keyword 'Atoms.SpeciesAndCoordinates', and the second column is the number of FNAN+SNAN. Then, the first and second neighboring atoms in each truncated cluster are determined by taking account of the distance between the central atom and neighboring atoms so that the number of FNAN+SNAN can be equivalent to the value provided by the second column. FNAN+SNAN may largely change when unit vectors are changed, leading to sudden change of the total energy as a function of lattice constant. The user definition of FNAN+SNAN is useful to avoid such a case.

# 24 MPI parallelization

For large-scale calculations, parallel execution by MPI is supported for parallel machines with distributed memories.

## 24.1 O($N$) calculation

When the O($N$) method is employed, it is expected that one can obtain a good parallel efficiency because of the inherent algorithm. A typical MPI execution is as follows:

```
% mpirun -np 4 openmx DIA512_DC.dat > dia512_dc.std &
```

The input file 'DIA512_DC.dat' found in the directory 'work' is for the SCF calculation (1 MD) of the diamond including 512 carbon atoms using the divide-conquer (DC) method. The speed-up ratio in comparison of the elapsed time per MD step is shown in Fig. 25 (a) as a function of the number of processes on a CRAY-XC30 (2.6 GHz/Xeon processors). We see that the parallel efficiency decreases as the number of processors increase, and the speed-up ratio at 128 CPUs is about 84. The decreasing efficiency is due to the decrease of the number of atoms allocated to one processor. So, the weight of other unparallelized parts such as disk I/O becomes significant. Moreover, it should be noted that the efficiency is significantly reduced in non-uniform systems in terms of atomic species and geometrical structure due to disruption of the road balance, while an algorithm is implemented to avoid the disruption. See also the subsections 'DC-LNO method' and 'Krylov subspace method' for further information on parallelization.

## 24.2 Cluster calculation

In the cluster calculation, a double parallelization is made for two loops: spin multiplicity and eigenstates, where the spin multiplicity is one for the spin-unpolarized and non-collinear calculation, and two for the spin-polarized calculation, respectively. The priority of parallelization is in order of spin multiplicity and eigenstates. OpenMX Ver. 3.9 employs ELPA [39] to solve the eigenvalue problem in the cluster calculation, which is a highly parallelized eigevalue solver. Either ELPA1 or ELPA2 can be chosen by the following keyword:

```
scf.eigen.lib      elpa1    # elpa1|elpa2, default=elpa1
```

The default choice is ELPA1. Our benchmark calculations suggest that ELPA1 and ELPA2 are comparable to each other with respect to the computational speed, while we do not show the benchmark calculations here. Figure 25 (b) shows the speed-up ratio as a function of processors in the elapsed time for a spin-polarized calculation of a single molecular magnet consisting of 148 atoms. The input file 'Mn12.dat' is found in the directory 'work'. It is found that the speed-up ratio is 11 and 17 using 32 and 64 processes, respectively.

## 24.3 Band calculation

In the band calculation, a triple parallelization is made for three loops: spin multiplicity, **k**-points, and eigenstates, where the spin multiplicity is one for the spin-unpolarized and non-collinear calculations,

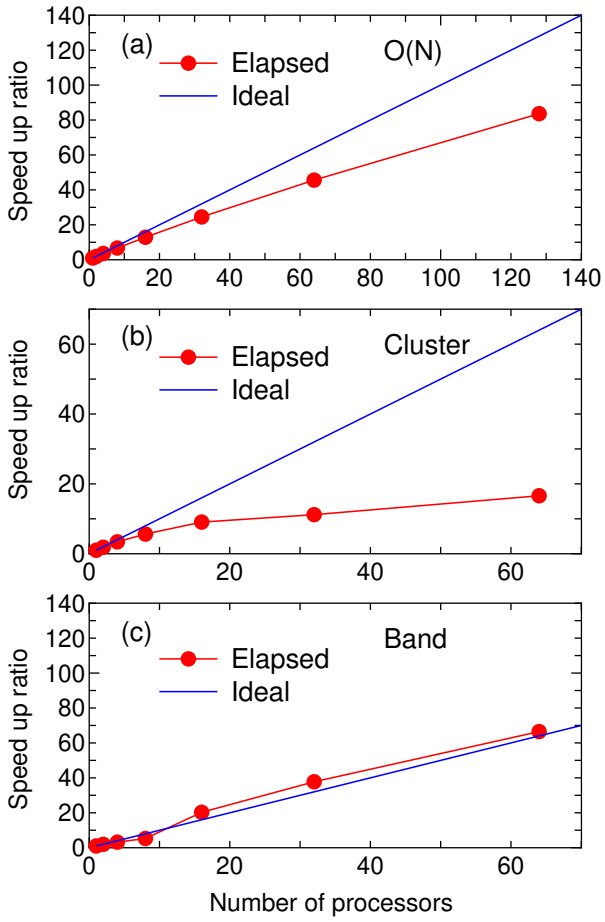



Figure 25: Speed-up ratio of the elapsed time per MD step in parallel calculations using MPI on a CRAY-XC30 (2.6 GHz Xeon processors) (a) for the carbon diamond including 512 atoms in the supercell by the DC method, (b) for a single molecular magnet consisting of 148 atoms by the cluster method, and (c) for the carbon diamond including 64 atoms in the super cell by the band method with 3×3×3 k-points. For comparison, a line which corresponds to the ideal speed-up ratio is also shown.

and two for the spin-polarized calculation, respectively. The priority of parallelization is in order of spin multiplicity, **k**-points, and eigenstates. In addition, when the number of processes used in the parallelization exceeds (spin multiplicity)×(the number of k-points), OpenMX uses an efficient way in which finding the Fermi level and calculating the density matrix are performed by just one diagonalization at each **k**-point. For the other cases, twice diagonalizations are performed at each **k**-point for saving the size of used memory in which the second diagonalization is performed to calculate the density matrix after finding the Fermi level. In Fig. 25 (c) we see a good speed-up ratio as a function of processes in the elapsed time for a spin-unpolarized calculation of carbon diamond consisting of 64 carbon atoms with 3×3×3 k-points. The input file 'DIA64_Band.dat' is found in the directory 'work'. In this case the spin multiplicity is one, and the number of k-points used for the actual calculation is (3*3*3-1)/2+1=14, since the **k**-points in the half Brillouin zone is taken into account for the collinear calculation, and the Γ-point is included when all the numbers of **k**-points for **a**-, **b**-, and

**c**-axes are odd. So it is found that the speed-up ratio exceeds the ideal one in the range of processes over 14, which means the algorithm in the parallelization is changed to the efficient scheme. As well as the cluster calculation, OpenMX Ver. 3.9 employs ELPA [39] to solve the eigenvalue problem in the band calculation, which is a highly parallelized eigevalue solver. Either ELPA1 or ELPA2 can be chosen by the following keyword:

```
scf.eigen.lib      elpa1    # elpa1|elpa2, default=elpa1
```

The default choice is ELPA1. Our benchmark calculations suggest that ELPA1 and ELPA2 are comparable to each other with respect to the computational speed, while we do not show the benchmark calculations here.

### 24.4 Fully three dimensional parallelization

OpenMX Ver. 3.9 supports a fully three dimensional parallelization for data distribution, while up to and including Ver. 3.6, the parallelization is made by a simple one-dimensional domain decomposition for **a**-axis of the unit cell for data distribution. Thus, users do not need to care about how unit cells are specified to achieve good road balancing. In OpenMX Ver. 3.9, a nearly equivalent parallel efficiency will be obtained without depending on choice of the unit cell vectors.

### 24.5 Maximum number of processors

Up to and including Ver. 3.6, the number of MPI processes that users can utilize for the parallel calculations is limited up to the number of atoms in the system. OpenMX Ver. 3.9 does not have the limitation. Even if the number of MPI processes exceeds the number of atoms, the MPI parallelization is efficiently performed. The capability may be useful especially for a calculation where the number of **k**-points is much larger than the number of atoms in the system.

# 25 MPI/OpenMP hybrid parallelization

The MPI/OpenMP hybrid parallel execution can be performed by

```
% mpirun -np 32 openmx DIA512-1.dat -nt 4 > dia512-1.std &
```

where '-nt' means the number of threads in each process managed by MPI. If '-nt' is not specified, then the number of threads is set to 1, which corresponds to the flat MPI parallelization. Since the parallelization of OpenMX Ver. 3.9 is largely changed from OpenMX Ver. 3.6, we do not have enough data to validate the hybrid parallelization compared to the flat MPI with respect to efficiency of computation and memory usage. However, our preliminary benchmark calculations imply that the hybrid parallelization seems to be efficient as for memory usage, while the computational efficiency seems to be comparable to each other.

# 26 Large-scale calculations

## 26.1 Conventional scheme

Using the conventional diagonalization method, OpenMX Ver. 3.9 is capable of performing geometry optimization for systems consisting of 1000 atoms if several hundreds processor cores are available. To demonstrate the capability, one can perform 'runtestL2' as follows:

```
% mpirun -np 128 openmx -runtestL2 -nt 4
```

Then, OpenMX will run with 7 test files, and compare calculated results with the reference results which are stored in 'work/large2_example'. The following is a result of 'runtestL2' performed using 640 MPI processes and 1 OpenMP threads on a Xeon cluster machine.

| | | | | |
|---|---|---|---|---|
| 1 | large2_example/C1000.dat | Elapsed time(s)= 777.60 | diff Utot= 0.000000007341 | diff Force= 0.000000008795 |
| 2 | large2_example/Fe1000.dat | Elapsed time(s)= 8181.70 | diff Utot= 0.000000002241 | diff Force= 0.000000011061 |
| 3 | large2_example/GRA1024.dat | Elapsed time(s)= 927.20 | diff Utot= 0.000000012903 | diff Force= 0.000000004981 |
| 4 | large2_example/Ih-Ice1200.dat | Elapsed time(s)= 445.88 | diff Utot= 0.000000000216 | diff Force= 0.000000001451 |
| 5 | large2_example/Pt500.dat | Elapsed time(s)= 2629.20 | diff Utot= 0.000000015832 | diff Force= 0.000000001879 |
| 6 | large2_example/R-TiO2-1050.dat | Elapsed time(s)= 844.58 | diff Utot= 0.000000002263 | diff Force= 0.000000001108 |
| 7 | large2_example/Si1000.dat | Elapsed time(s)= 658.53 | diff Utot= 0.000000000404 | diff Force= 0.000000000908 |

Total elapsed time (s) 14464.69

The quality of all the calculations is at a level of production run where double valence plus a single polarization functions are allocated to each atom as basis functions. Except for 'Pt500.dat', all the systems include more than 1000 atoms, where the last number of the file name implies the number of atoms for each system, and the elapsed time implies that geometry optimization for systems consisting of 1000 atoms is possible if several hundreds processor cores are available. The input files used for the calculations and the output files are found in the directory 'work/large2_example'. The following information is compiled from the output files.

| No. | Input file | SCF steps | Elapsed time(s/SCF/spin) | Dimension |
|---|---|---|---|---|
| 1 | large2_example/C1000.dat | 53 | 14.7 | 13000 |
| 2 | large2_example/Fe1000.dat | 408 | 10.0 | 13000 |
| 3 | large2_example/GRA1024.dat | 72 | 12.9 | 13312 |
| 4 | large2_example/Ih-Ice1200.dat | 57 | 7.8 | 9200 |
| 5 | large2_example/Pt500.dat | 161 | 16.3 | 12500 |
| 6 | large2_example/R-TiO2-1050.dat | 38 | 22.2 | 15750 |
| 7 | large2_example/Si1000.dat | 45 | 14.6 | 13000 |

The dimension of the Kohn-Sham Hamiltonian is of the order of 10000, and the elapsed time per SCF step is around 15 seconds for all the systems, implying that the difference in the total elapsed time mainly comes from the difference in the SCF iterations to achieve the SCF convergence of 10e-10 (Hartree) for the band energy.

## 26.2 Combination of the O(N) and conventional schemes

Although the O($N$) methods can treat large-scale systems consisting of more than 1000 atoms, a serious problem is that information about wave functions is lost in the O($N$) methods implemented in OpenMX. A simple way of obtaining wave functions and the corresponding eigenvalues for the large-scale systems is firstly to employ the O($N$) methods to obtain a self-consistent charge density,

and then is to just once diagonalize using the conventional diagonalization method under the self-consistent charge density to obtain full wave functions. As an illustration of this procedure, we show a large-scale calculation of a multiply connected carbon nanotube (MCCN) consisting of 564 carbon atoms. First, the SCF calculation of a MCCN was performed using the O($N$) Krylov subspace method and 16 CPU cores of a 2.6 GHz Xeon, where C5.0-s2p1 (basis function), 130 Ryd (scf.energycutoff), 1.0e-7 (scf.criterion), 6.5 Å (orderN.HoppingRanges), 'orderN.KrylovH.order=400', and RMM-DIISK (mixing scheme) were used. The input file is 'MCCN.dat' in the directory 'work'. Figure 26 shows the norm of residual charge density in Fourier space as a function of SCF steps. We see that 56 SCF steps is enough to obtain convergent charge density for the system, where the computational time was about seven minutes. After that, the following keywords were set in

```
scf.maxIter                1
scf.EigenvalueSolver    Band
scf.Kgrid              1 1 1
scf.restart               on
MO.fileout                on
num.HOMOs                  2
num.LUMOs                  2
MO.Nkpoint                 1
<MO.kpoint
  0.0  0.0  0.0
MO.kpoint>
```

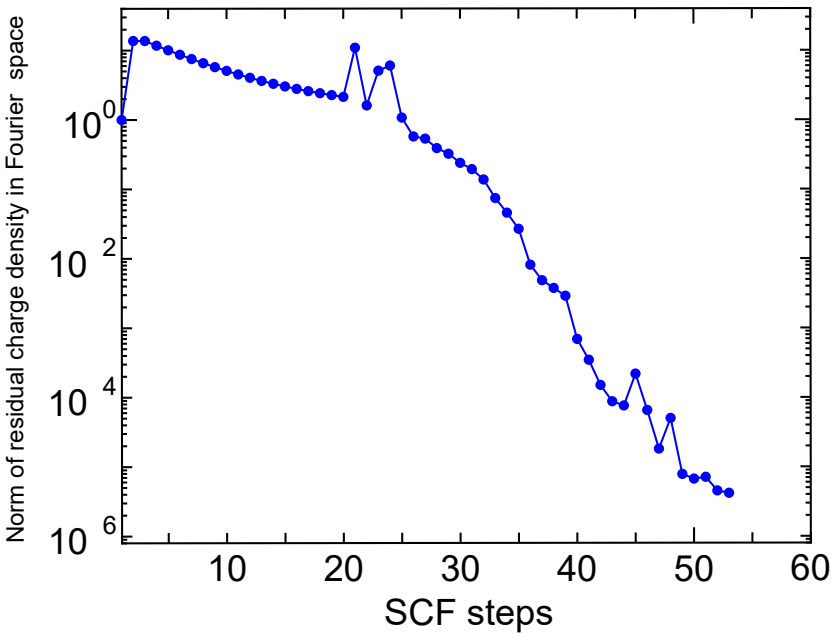



Figure 26: Norm of residual charge density in Fourier space as a function of SCF steps for a multiply connected carbon nanotube (MCCN) consisting of 564 carbon atoms. The input file is 'MCCN.dat' in the directory 'work'.

Then we calculated the same system in order to obtain wave functions using 16 CPU cores of a 2.6 GHz Xeon machine, where the computational time was about 2 minutes. Figure 27 shows isosurface maps of the HOMO and LUMO ($\Gamma$-point) of MCCN calculated by the above procedure. Although the difference between the O($N$) method and the conventional diagonalization scheme in the computational time is not significant in this example, the procedure will be useful for larger system including more than several thousands atoms.

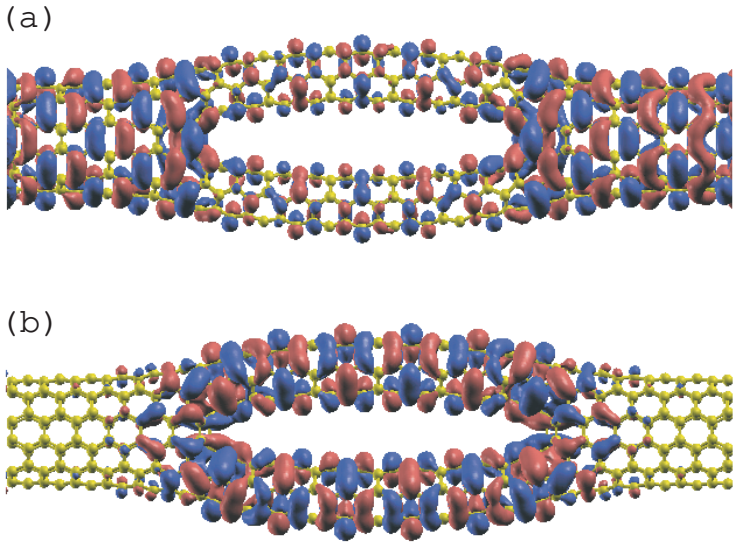



Figure 27: Isosurface map of (a) the highest occupied molecular orbital (HOMO) and (b) the lowest unoccupied molecular orbital (LUMO) of a multiply connected carbon nanotube (MCCN) consisting of 564 carbon atoms, where |0.005| was used as an isovalue of the molecular orbital.

# 27 Electric field

It is possible to apply a uniform external electric field given by a sawtooth waveform during the SCF calculation and the geometry optimization. For example, when an electric field of 1.0 GV/m ($10^9$ V/m) is applied along the **a**-axis, please specify the keyword 'scf.Electric.Field' in your input file as follows:

```
scf.Electric.Field   1.0 0.0 0.0   # default=0.0 0.0 0.0 (GV/m)
```

The sign of electric field is taken as that applied to electrons. If the uniform external electric field is applied to a periodic bulk system without vacuum region, discontinuities of the potential are introduced, which may cause numerical instability. On the other hand, for molecular systems, the discontinuities are located in the vacuum region, indicating that numerical instability may not be induced.

As an illustration of the electric field, changes of total charge in a nitrobenzene molecule induced by the electric field are shown in Fig. 28. We can see that a large charge transfer takes place among oxygens in -$NO_2$, para-carbon atom, and para-hydrogen atom. The input file is 'Nitro_Benzene.dat' in the directory 'work'. See also Section 63 'Analysis of difference in two Gaussian cube files' as for the difference charge maps shown in Fig. 28.

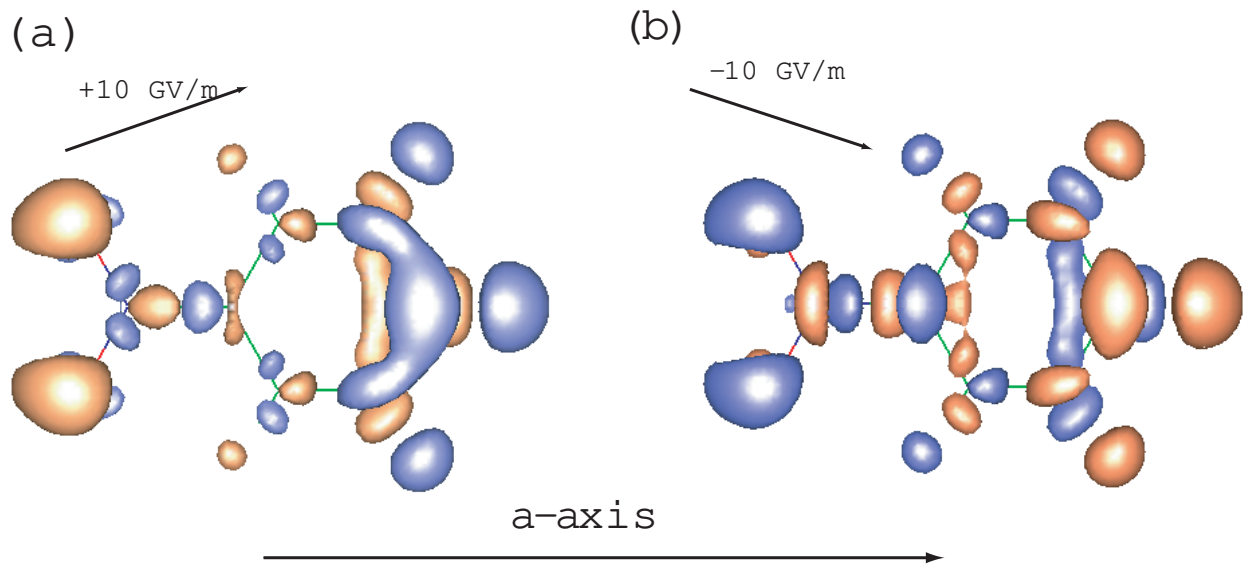



Figure 28: Difference in the total charge density of a nitrobenzene molecule between the zero-bias voltage and applied bias voltage along the **a**-axis of (a) 10 GV/m, and (b) -10 GV/m, where orange and blue colors mean the increase and decrease of charge density. Tilted arrows depict the slope of applied electric fields. The input file is 'Nitro_Benzene.dat' in the directory 'work'.

# 28 Charge doping

The following keyword is available for both the electron and hole dopings.

```
scf.system.charge      1.0      # default=0.0
```

The plus and minus signs correspond to hole and electron dopings, respectively. A partial charge doping is also possible. The excess charge given by the keyword 'scf.system.charge' is compensated by a uniform background opposite charge, since FFT is used to solve Poisson's equation in OpenMX. Therefore, if you compare the total energy between different charged states, a careful treatment is required, because additional electrostatic interactions induced by the background charge are included in the total energy. Note that in Sec. 58, a proper way of treating charged and isolated systems is discussed.

As an example, we show spin densities of hole doped, neutral, and electron doped (5,5) carbon nanotubes with a finite length of 14 Å in Fig. 29. The neutral and electron doped nanotubes possess the total spin moment of 1.0 and 2.2, while the total spin moment almost disappears in the hole doped nanotube. We can see that the spin polarization takes place at the edges of the neutral and electron doped nanotubes due to dangling bonds of edge regions.

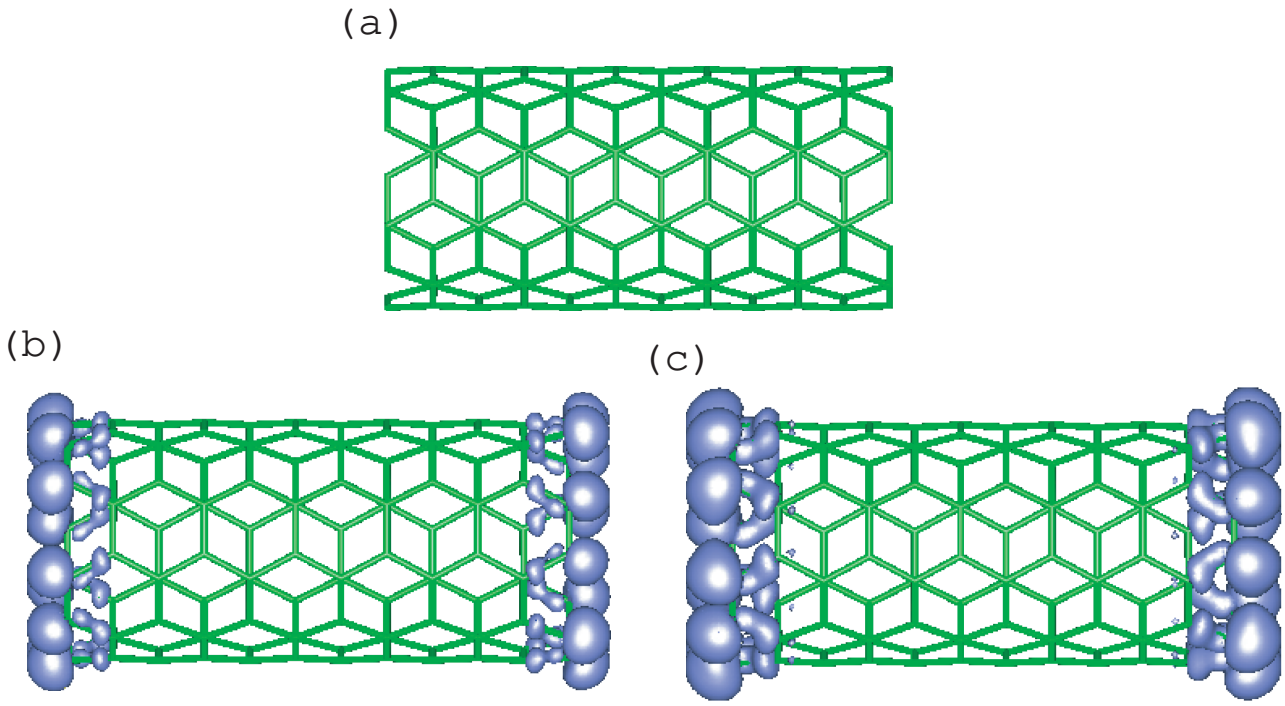



Figure 29: Spin densities of (a) four hole doped, (b) neutral, and (c) four electron doped (5,5) carbon nanotubes with a finite length of 14 Å. The input file is 'Doped_NT.dat' in the directory 'work'.

## 29 Virtual atom with fractional nuclear charge

It is possible to treat a virtual atom with fractional nuclear charge by using a pseudopotential with the corresponding fractional nuclear charge. The pseudopotential for the virtual atom can be generated by ADPACK. The relevant keywords in ADPACK are given by

```
AtomSpecies             6.2
total.electron          6.2
valence.electron        4.2
<occupied.electrons
 1    2.0
 2    2.0   2.2
occupied.electrons>
```

The above example is for a virtual atom on the way of carbon and nitrogen atoms. Also, it is noted that basis functions for the pseudopotential of the virtual atom must be generated for the virtual atom with the same fractional nuclear charge, since the atomic charge density stored in *.pao is used to make the neutral atom potential.

As an illustration, the DOS of $C_{7.8}N_{0.2}$ calculated using the method is shown in Fig. 30. The input file is 'DIA8-VA.dat' which can be found in the directory 'work'. In the calculation, one of eight carbon atoms in the unit cell was replaced by a virtual atom with an effective nuclear charge of 4.2, which corresponds to a stoichiometric compound of $C_{7.8}N_{0.2}$.

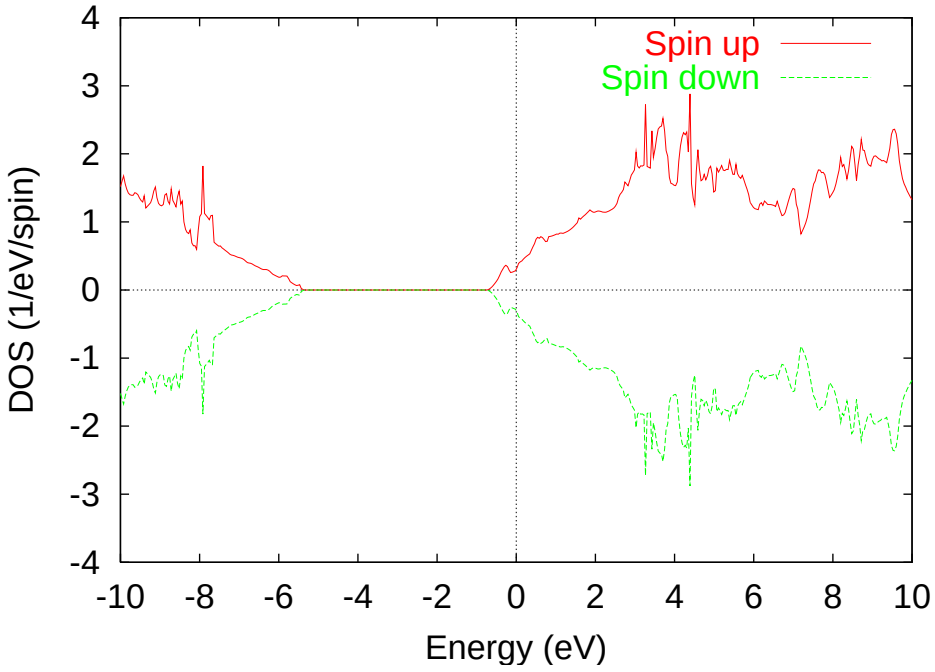



Figure 30: Density of states (DOS) of $C_{7.8}N_{0.2}$ calculated with a pseudopotential of the virtual atom. The input file used for the calculation is 'DIA8-VA.dat' which can be found in the directory 'work'.

# 30 LCAO coefficients

It is possible to analyze LCAO coefficients in both the cluster and band calculations. In the cluster calculation, if a keyword 'level.of.fileout'' is set in '2', the LCAO coefficients are added into a file '*System.Name*.out'. As an example, LCAO coefficients of 'Methane.dat' discussed in the Section 'Test calculation' are shown below:

```
***********************************************************
***********************************************************
  Eigenvalues (Hartree) and Eigenvectors for SCF KS-eq.
***********************************************************
***********************************************************

   Chemical Potential (Hartree) =   0.00000000000000
   HOMO =  4

   LCAO coefficients for up (U) and down (D) spins


                          1 (U)     2 (U)     3 (U)     4 (U)     5 (U)     6 (U)
                        -0.69899  -0.41525  -0.41525  -0.41524   0.21215   0.21215

   1   C 0 s             0.69137  -0.00000   0.00000   0.00000   0.00000   0.00000
         0 px            0.00000  -0.10055   0.63544   0.00033  -0.68649  -1.00467
         0 py            0.00000   0.00028  -0.00029   0.64331   0.00000  -0.00001
         0 pz           -0.00000   0.63544   0.10055  -0.00023  -1.00467   0.68649
   2   H 0 s             0.12870   0.05604  -0.35474  -0.25425  -0.59781  -0.87489
   3   H 0 s             0.12870  -0.35475  -0.05627   0.25420  -0.87488   0.59781
   4   H 0 s             0.12870   0.35497   0.05604   0.25393   0.87488  -0.59781
   5   H 0 s             0.12870  -0.05626   0.35497  -0.25388   0.59781   0.87488

                          7 (U)     8 (U)
                        0.21223   0.24739

   1   C 0 s             0.00000   1.90847
         0 px            0.00000   0.00000
         0 py           -1.21683  -0.00000
         0 pz           -0.00000   0.00000
   2   H 0 s            -0.74926  -0.76083
   ......
   ....
```

In bulk calculations, if a keyword 'MO.fileout' is set to 'ON', LCAO coefficients at k-points which are specified by the keyword 'MO.kpoint' are output into a file '*System.Name*.out'. For cluster calculations, 'level.of.fileout' should be 2 in order to output LCAO coefficients. But, for band calculations, the relevant keyword is 'MO.fileout' rather than 'level.of.fileout'.

# 31 Molecular orbitals

Molecuar or crystal Kohn-Sham orbitals can be output in the Gaussian cube format, and thereby visualized by many software such as VESTA [103] and XCrySDen [105]. The relevant keywords are given by

```
MO.fileout          on    # on|off, default=off
num.HOMOs            1    # default=2
num.LUMOs            1    # default=2
MO.Nkpoint           2    # default=1

<MO.kpoint
  0.0  0.0  0.0
  0.5  0.0  0.0
MO.kpoint>
```

When you want to generate the cube files, please swtich on the keyword 'MO.fileout'. The numbers of the highest occupied molecular (crystal) orbitals (HOMOs) and the lowest occupied molecular (crystal) orbitals (LUMOs) to be output can be specified by the keywords 'num.HOMOs' and 'num.LUMOs', respectively. In case of a band calculation, the **k**-points at which the HOMOs and LUMOs are calculated are specified by the keywords: 'MO.Nkpoint' and 'MO.kpoint'. The keyword 'MO.Nkpoint' gives the number of **k**-points at which the HOMOs and LUMOs are calculated, and by the keyword 'MO.kpoint' you can specifiy the **k**-points explicitly as shown above, where the specification is made based on the reciprocal vectors for the unit cell vectors given by 'Atoms.UnitVectors'. The output cube files are summarized as below:

**Cluster cases:**

If 'MO.fileout=ON' and 'scf.EigenvalueSolver=Cluster', the following files are also generated:

- *System.Name*.homo0_0.cube, *System.Name*.homo0_1.cube, ...

  The HOMOs are output in the Gaussian cube format. The first number below 'homo' means a spin state (up=0, down=1). The second number specifies the eigenstates, i.e., 0, 1, and 2 correspond to HOMO, HOMO-1, and HOMO-2, respectively, whose number is specified by the keyword 'num.HOMOs'.

- *System.Name*.lumo0_0.cube, *System.Name*.lumo0_1.cube, ...

  The LUMOs are output in the Gaussian cube format. The first number below 'lumo' means a spin state (up=0, down=1). The second number specifies the eigenstates, i.e., 0, 1, and 2 correspond to LUMO, LUMO+1, and LUMO+2, respectively, whose number is specified by the keyword 'num.LUMOs'.

**Bulk cases:**

If 'MO.fileout=ON' and 'scf.EigenvalueSolver=Band', the following files are also generated:

- *System.Name*.homo0_0_0_r.cube, *System.Name*.homo1_0_1_r.cube, ... *System.Name*.homo0_0_0_i.cube, *System.Name*.homo1_0_1_i.cube, ...

  The HOMOs are output in the Gaussian cube format. The first number below 'homo' means the k-point number, which is specified by the keyword 'MO.kpoint'. The second number is a spin state (up=0, down=1). The third number specifies the eigenstates, i.e., 0, 1, and 2 correspond to HOMO, HOMO-1, and HOMO-2, respectively, whose number is specified by the keyword 'num.HOMOs'. The 'r' and 'i' mean the real and imaginary parts of the wave function.

- *System.Name*.lumo0_0_0_r.cube, *System.Name*.lumo1_0_1_r.cube, ... *System.Name*.lumo0_0_0_i.cube, *System.Name*.lumo1_0_1_i.cube, ...

  The LUMOs are output in the Gaussian cube format. The first number below 'lumo' means the k-point number, which is specified in the keyword 'MO.kpoint'. The second number is a spin state (up=0, down=1). The third number specifies the eigenstates, i.e., 0, 1, and 2 correspond to LUMO, LUMO+1, and LUMO+2, respectively, whose number is specified by the keyword 'num.LUMOs'. The 'r' and 'i' mean the real and imaginary parts of the wave function.

As an example, Fig. 31 show the HOMO and LUMO of a valorphin molecule.

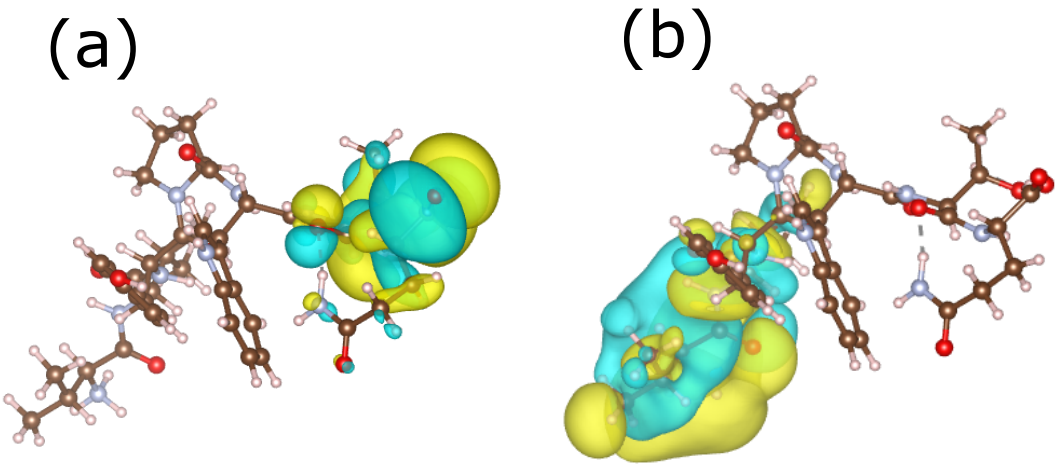



Figure 31: (a) HOMO of a valorphin molecule, and (b) LUMO of a valorphin molecule. The figures were visualized by VESTA [103]. The input file is 'Valorphin_MO.dat' in the directory 'work'.

# 32 Charge analysis

Although it is a somewhat ambiguous issue to assign effective charge to each atom, OpenMX provides three schemes, Mulliken charge analysis, Voronoi charge analysis, and electro-static potential (ESP) fitting method, to analyze the charge state of each atom.

## 32.1 Mulliken charge

The Mulliken charges are output in '*System.Name*.out' by default as shown in Section 'Test calculation'. In addition to the Mulliken charge projected to each atom, you can also find a decomposed Mulliken charge to each orbital in '*System.Name*.out'. The result stored in '*System.Name*.out' for a methane molecule is as follows:

```
Decomposed Mulliken populations

  1    C          Up spin      Down spin     Sum           Diff
          multiple
s           0    0.598003833  0.598003833   1.196007667   0.000000000
 sum over m      0.598003833  0.598003833   1.196007667   0.000000000
 sum over m+mul  0.598003833  0.598003833   1.196007667   0.000000000
px          0    0.588514081  0.588514081   1.177028163   0.000000000
py          0    0.588703212  0.588703212   1.177406424   0.000000000
pz          0    0.588514081  0.588514081   1.177028162   0.000000000
 sum over m      1.765731375  1.765731375   3.531462749   0.000000000
 sum over m+mul  1.765731375  1.765731375   3.531462749   0.000000000

  2    H          Up spin      Down spin     Sum           Diff
          multiple
s           0    0.409066346  0.409066346   0.818132693   0.000000000
 sum over m      0.409066346  0.409066346   0.818132693   0.000000000
 sum over m+mul  0.409066346  0.409066346   0.818132693   0.000000000

  3    H          Up spin      Down spin     Sum           Diff
          multiple
s           0    0.409065912  0.409065912   0.818131824   0.000000000
 sum over m      0.409065912  0.409065912   0.818131824   0.000000000
 sum over m+mul  0.409065912  0.409065912   0.818131824   0.000000000

 .......
 ....
```

As you can see, the Mulliken charges are decomposed for all the orbitals. There are two kind of summations in this decomposition. One of the summations is 'sum over m' which means a summation over magnetic quantum number for each multiple orbital. The second summation is 'sum over m+mul'

which means a summation over both magnetic quantum number and orbital multiplicity, where "multiple" means a number to specify a radial wave function. Therefore, Mulliken charges are decomposed to contributions of all the orbitals.

## 32.2 Voronoi charge

Voronoi charge of each atom is calculated by integrating electron and spin densities in a Voronoi polyhedron. The Voronoi polyhedron is constructed from smeared surfaces which are defined by a Fuzzy cell partitioning method [69]. It should be noted that this Voronoi analysis gives often overestimated or underestimated charge, since Voronoi polyhedron is determined by only the structure without taking account of atomic radius. If you want to calculate Voronoi charge, specify the following keyword 'Voronoi.charge' in your input file:

```
Voronoi.charge      on       # on|off, default = off
```

In case of a methane molecule, the following Voronoi charges are output to '*System.Name*.out'.

```
***********************************************************
***********************************************************
                    Voronoi charges
***********************************************************
***********************************************************

  Sum of Voronoi charges for up    =  4.000000290723
  Sum of Voronoi charges for down  =  4.000000290723
  Sum of Voronoi charges for total =  8.000000581446

  Total spin magnetic moment (muB) by Voronoi charges  =  0.000000000000

                    Up spin      Down spin     Sum           Diff       Voronoi Volume (Ang.^3)
      Atom=   1   1.129270484  1.129270484   2.258540969   0.000000000   2.355421391
      Atom=   2   0.717682452  0.717682452   1.435364903   0.000000000  62.245579466
      Atom=   3   0.717682452  0.717682452   1.435364903   0.000000000  62.245579466
      Atom=   4   0.717682452  0.717682452   1.435364903   0.000000000  62.245579466
      Atom=   5   0.717682451  0.717682451   1.435364903   0.000000000  62.245579466
```

Clearly, we see that carbon atom (Atom=1) and hydrogen atoms (Atom=2-5) are charged positively and negatively, respectively, which apparently contradicts a usual chemical sense. However, the Voronoi analysis could be a useful and complementary information for a bulk system with a closed pack structure. Also, the Voronoi volume being a supplemental information will be useful to analyze local structure for bulk systems.

## 32.3 Electro-static potential fitting

For small molecular systems, the electro-static potential (ESP) fitting method [109, 110, 111] is useful to determine an effective charge of each atom, while the ESP fitting method cannot be applied for large molecules and bulk systems, since there are not enough sampling points for atoms far from surface areas in the ESP fitting method. In the ESP fitting method an effective net point charge on each atom is determined by a least square method with constraints so that the sum of the electro-static potential by effective point charges can reproduce electro-static potential calculated by the DFT calculation as

much as possible. The ESP fitting charge is calculated by the following two steps:

**(1) SCF calculation**

After finishing a usual SCF calculation, you have two output files:

```
System.Name.out
System.Name.vhart.cube
```

There is no additional keyword to generate the two files which are default output files by the SCF calculation, while the keyword 'level.of.fileout' should be 1 or 2.

**(2) ESP fitting charge**

Let us compile a program code for calculating the ESP fitting charge. Move to the directory 'source' and then compile as follows:

```
% make esp
```

When the compilation is completed normally, then you can find an executable file 'esp' in the directory 'work'. The ESP fitting charge can be calculated from two files '*System.Name*.out' and '*System.Name*.vhart.cube' using the program 'esp'. For example, you can calculate them for a methane molecule shown in the Section 'Input file' as follows:

```
% ./esp met -c 0 -s 1.4 2.0
```

Then, it is enough to specify the file name without the file extension, however, two files 'met.out' and 'met.vhart.cube' must exist in the directory 'work'. The options '-c' and '-s' are key parameters to specify a constraint and scale factors. You can find the following statement in the header part of a source code 'esp.c':

```
-c      constraint parameter
        '-c 0' means charge conservation
        '-c 1' means charge and dipole moment conservation
-s      scale factors for vdw radius
        '-s 1.4 2.0' means that 1.4 and 2.0 are 1st and 2nd scale factors
```

In the ESP fitting method, we support two constraints, charge conservation and, charge and dipole moment conservation. Although the latter can reproduce charge and dipole moment calculated by the DFT calculation, it seems that the introduction of the dipole moment conservation gives often physically unacceptable point charges especially for a relatively large molecule. Thus, we would like to recommend the former constraint. The sampling points are given by the grids in real space between two shells of the first and second scale factors times van der Waals radii [112]. In the above example, 1.4 and 2.0 correspond to the first and second scale factors. The calculated result appears in the standard output (your display) as follows:

```
 % ./esp met -c 0 -s 1.4 2.0

*********************************************************************
*********************************************************************
 esp: effective charges by a ESP fitting method
 Copyright (C), 2004, Taisuke Ozaki
 This is free software, and you are welcome to
 redistribute it under the constitution of the GNU-GPL.
*********************************************************************
*********************************************************************

Constraint: charge
Scale factors for vdw radius    1.40000    2.00000
Number of grids in a van der Waals shell = 28464
Volume per grid =    0.0235870615 (Bohr^3)
Success

  Atom=   1  Fitting Effective Charge= -0.93558216739
  Atom=   2  Fitting Effective Charge=  0.23389552572
  Atom=   3  Fitting Effective Charge=  0.23389569182
  Atom=   4  Fitting Effective Charge=  0.23389535126
  Atom=   5  Fitting Effective Charge=  0.23389559858

  Magnitude of dipole moment    0.0000015089 (Debye)
  Component x y z     0.0000003114   -0.0000002455   -0.0000014558
RMS between the given ESP and fitting charges (Hartree/Bohr^3)= 0.096515449505
```

# 33 Natural population analysis

In the natural bond orbital (NBO) method developed by Weinhold [6], atomic population (or atomic charge) is calculated based on atomically localized orbitals, natural atomic orbitals (NAO), which is referred to as natural population (NP). OpenMX supports the NP analysis. Especially, for large-sized calculation models, one can efficiently calculate and analyze NPs by selecting target atoms, which is an original scheme based on a truncated cluster method [7]. In OpenMX, the NP calculation is carried out after the SCF calculation, and results of NP analysis are shown in a standard output. The way of NP calculation is as follows:

**(a) Case of small-sized calculation (scf.EigenvalueSolver = Cluster)**

The NP calculation is supported by the following keyword :

```
NBO.switch   on1
```

This type of NP calculation is supported in case of 'scf.EigenvalueSolver = Cluster', and carried out by using a full-sized density matrix. As a sample of this type of NP calculation, users can refer to an input and output files for an ethylene carbonate (EC) molecule including 10 atoms, 'EC_NAO.dat' and 'EC_NAO.std', which are stored in a directory 'work/nbo_example'.

**(b) Case of large-size calculation (scf.EigenvalueSolver = Krylov)**

The NAO calculation for a large-sized model carried out with 'NBO.switch = on1' will be sometimes hampered by memory shortage due to the large-sized density matrix. For the NP calculation of a large-sized model, the following option for the keyword 'NBO.switch' is available

```
NBO.switch   on2
```

The option 'on2' has to be used with the O(N) Krylov subspace calculation (scf.EigenvalueSolver = Krylov). After the O(N) SCF calculation, the NP calculation can be efficiently performed by selecting a few atoms you are interested in. When a large-sized model is treated, it is not always necessary for one to calculate NPs for all atoms, that is, in most cases, NP analysis on a few atoms in local regions is enough to investigate important events such as chemical reactions. In OpenMX, target atoms for the NP analysis can be selected by the following keywords:

```
NBO.Num.CenterAtoms     5
<NBO.CenterAtoms
  269
  304
  323
  541
  574
NBO.CenterAtoms>
```

By the keyword 'NBO.Num.CenterAtoms' , one designates the number of target atoms. Between '<NBO.CenterAtoms' and 'NBO.CenterAtoms>', one describes serial indexes of target atoms, which are specified by the keyword 'Atoms.SpeciesAndCoordinates'. The keyword 'NBO.CenterAtoms' is

valid only if 'on2' is chosen for 'NBO.switch'. In case of 'on1' for the keyword 'NBO.switch', NPs for all atoms will be calculated regardless of the specification of the keyword 'NBO.CenterAtoms'. As a sample of the NP calculation for a large-sized model, you can refer to an input and output files for amorphous SiO2 bulk system (648 atoms), 'SiO2_NAO.dat' and 'SiO2_NAO.std', which are stored in the directory 'work/nbo_example'.

**(c) Parameter for NAO calculation: NAO.threshold**

The NAO calculation has a process of distinguishing occupied and Rydberg (low-occupied) NAOs. The criterion of the distinction is given by the keyword 'NAO.threshold', by which one designates the number of electrons per spin orbital. The default value of this parameter is 0.85, with which most of NAO calculations are executed normally. If the obtained number of occupied NAOs is abnormal, you may need to adjust the value of 'NAO.threshold' .

**(d) Example**

Here the result of NP calculation for the EC molecule is shown. First, NPs for all the atoms and summation of those NPs are output in the standard output as follows:

```
   1   O :   6.46917105
   2   C :   4.09908587
   3   C :   4.09909317
   4   O :   6.46902031
   5   C :   3.14623972
   6   O :   6.50714720
   7   H :   0.80250093
   8   H :   0.80249967
   9   H :   0.80262024
  10   H :   0.80262185
------------------------
  Total  :  34.00000000

 ## Global atom num.: 1 ( O ) / NP =   6.4692
-------------------------------------------------------------------------------
 NP in NAO         0.0013  1.6803  0.0046  1.6731  0.0055  1.3000  0.0073  1.7972
 Energy (Hartree)  1.2529 -0.7196  0.4452 -0.3208  0.7108 -0.2935  0.4327 -0.3005
-------------------------------------------------------------------------------
 1 s              -1.8000  1.2493 -0.0776  0.1255 -0.0125 -0.0027  0.0000  0.0000
 2 s               1.9201  0.0014  0.0530 -0.0030  0.0163  0.0015 -0.0000 -0.0000
 1 px             -0.4312  0.1031 -0.6568  1.0499  0.0431  0.0115 -0.0000 -0.0000
 2 px              0.1913 -0.0062  1.7251  0.0215 -0.0500 -0.0052  0.0000  0.0000
 1 py             -0.1040 -0.0056  0.0371  0.0110 -2.1699  1.0361 -0.0000 -0.0000
 2 py              0.0573  0.0025 -0.0493 -0.0052  3.4499  0.1576  0.0000  0.0000
 1 pz              0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.6206  1.0131
 2 pz             -0.0000 -0.0000  0.0000  0.0000  0.0000  0.0000  1.7312  0.0267
 ...
 ..
```

After the NPs are shown for all the atoms, NPs and energy levels of NAOs on each atom are shown in the first and second lows, respectively, and followed by LCPAO coefficients for the corresponding NAO. In the near future, the calculation and analysis functions of NBO will be also supposed to be implemented in OpenMX.

# 34 Non-collinear DFT

A fully unconstrained non-collinear density functional theory (DFT) is supported including the spin-orbit coupling (SOC) [8, 9, 10, 11, 16]. When the non-collinear DFT is performed, the following option for the keyword 'scf.SpinPolarization' is available.

```
scf.SpinPolarization        NC        # On|Off|NC
```

When the option 'NC' is specified, wave functions are expressed by a two component spinor. An initial spin orientation of each site is given by

```
<Atoms.SpeciesAndCoordinates           # Unit=Ang
  1  Mn    0.00000   0.00000   0.00000   8.0  5.0  45.0 0.0 45.0 0.0  1 on
  2  O     1.70000   0.00000   0.00000   3.0  3.0  45.0 0.0 45.0 0.0  1 on
Atoms.SpeciesAndCoordinates>


 1:    sequential serial number
 2:    species name
 3:    x-coordinate
 4:    y-coordinate
 5:    z-coordinate
 6:    initial occupation for up spin
 7:    initial occupation for down spin
 8:    Euler angle, theta, of the magnetic field for spin magnetic moment
 9:    Euler angle, phi, of the magnetic field for spin magnetic moment
       Also, the 8th and 9th are used to generate the initial non-collinear
       spin charge distribution
10:    the Euler angle, theta, of the magnetic field for orbital magnetic moment
11:    the Euler angle, phi, of the magnetic field for orbital magnetic moment
12:    switch for the constraint schemes specified by the keywords
       'scf.Constraint.NC.Spin', 'scf.NC.Zeeman.Orbital' and 'scf.NC.Zeeman.Orbital'.
       '1' means that the constraint is applied, and '0' no constraint.
13:    switch for enhancement of orbital polarization in the LDA+U method,
       'on' means that the enhancement is made, 'off' no enhancement.
```

The initial Euler angles, $\theta$ and $\phi$, for orientation of the spin and orbital magnetic moment are given by the 8th and 9th columns, and 10th and 11th columns, respectively. The 12th column is a switch for a constraint scheme that a constraint (penalty or Zeeman) functional to the spin and orbital orientation is added on each site, where '1' means that the constraint functional is added, and '0' means no constraint. For the details of the constraint DFT for the spin orientation, see Sec. 38 'Constraint DFT for non-collinear spin orientation'. The final 13th column is a switch for enhancement of orbital polarization in the LDA+U method, 'on' means that the enhancement is made, 'off' no enhancement. Figure 32 shows the spin orientation in a MnO molecule calculated by the non-collinear DFT. You can follow the calculation using an input file 'Mol_MnO_NC.dat' in the directory 'work'. To visualize the spin orientation in real space, two files are generated:

```
System.Name.nc.xsf
System.Name.ncsden.xsf
```

where *System.Name* means 'System.Name' you specified. Two files '*System.Name*.nc.xsf' and '*System.Name*.ncsden.xsf' store a projected spin orientation to each atom by Mulliken analysis and the spin orientation on real space grids in a vector file format (XSF) supported by XCrySDen. Both the files can be visualized using 'Display→ Forces' in XCrySDen as shown in Fig. 32.

The spin moment and Euler angles of each atom, which are calculated by Mulliken analysis, are found in the file '*System.Name*.out' as follows:

```
*************************************************************
*************************************************************
                  Mulliken populations
*************************************************************
*************************************************************

  Total spin moment (muB)   4.998503442   Angles (Deg) 44.991211196   0.000000000

                Up       Down       Sum       Diff        theta       phi
 1   Mn    9.59803   4.76902   14.36705   4.82901    44.99208    0.00000
 2    O    3.40122   3.23173    6.63295   0.16949    44.96650   -0.00000
```

Also it should be noted that it is difficult to achieve a self consistent field in the non-collinear DFT more than the collinear DFT calculation, since there are many minima, having almost comparable energy, in the spin orientation space, while the constraint DFT is useful for such a case.

In the non-collinear DFT, the inclusion of spin-orbit coupling is supported, while it is not supported for the collinear DFT. See also the Section 'Relativistic effects' for the issue.

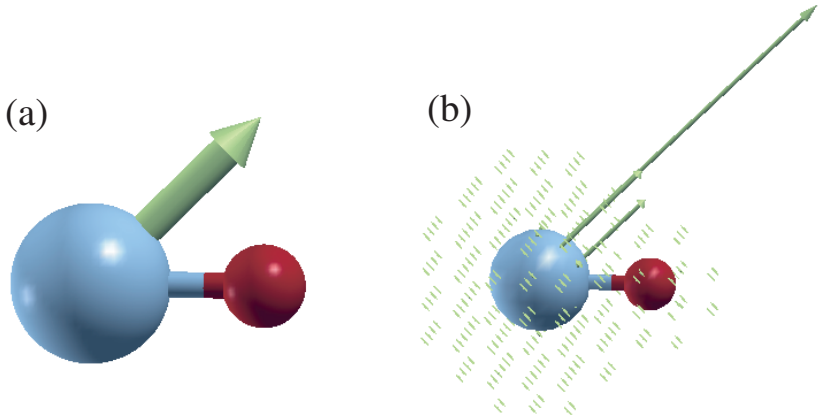



Figure 32: Spin orientation in (a) a projected form on each atom and (b) a real space representation of a MnO molecule calculated by the non-collinear DFT. The figures were visualized by 'Display→ Forces' in XCrySDen. The input file is 'Mol_MnO_NC.dat' in the directory 'work'.

# 35 Relativistic effects

Relativistic effects can be incorporated by fully relativistic and scalar relativistic pseudopotentials. In the fully relativistic treatment, the spin-orbit coupling is included in addition to kinematic relativistic effects (Darwin and mass velocity terms). On the other hand, the spin-orbit coupling is averaged in the scalar relativistic treatment. Although the scalar relativistic treatment can be incorporated in both the collinear and non-collinear DFT calculations, the fully relativistic treatment is supported for only the non-collinear DFT in OpenMX.

## 35.1 Fully relativistic

The fully relativistic effects including the spin-orbit coupling within the pseudopotential scheme can be included in the non-collinear DFT calculations [12, 32, 16], while the inclusion of the spin-orbit coupling is not supported in the collinear DFT calculation. The inclusion of fully relativistic effects is made by the following two steps:

**(1) Making of j-dependent pseudopotentials**

First, you are requested to generate j-dependent pseudopotentials using ADPACK. For your convenience, the j-dependent pseudopotentials are available for many elements in the database Ver. 2019 [149]. The details how to make the j-dependent pseudopotential are found in the manual of ADPACK.

**(2) SCF calculation**

If you specify j-dependent pseudopotentials in the specification of '<Definition.of.Atomic.Species', it is possible to include spin-orbit coupling by the following keyword 'scf.SpinOrbit.Coupling':

```
scf.SpinOrbit.Coupling      on          # On|Off, default=off
```

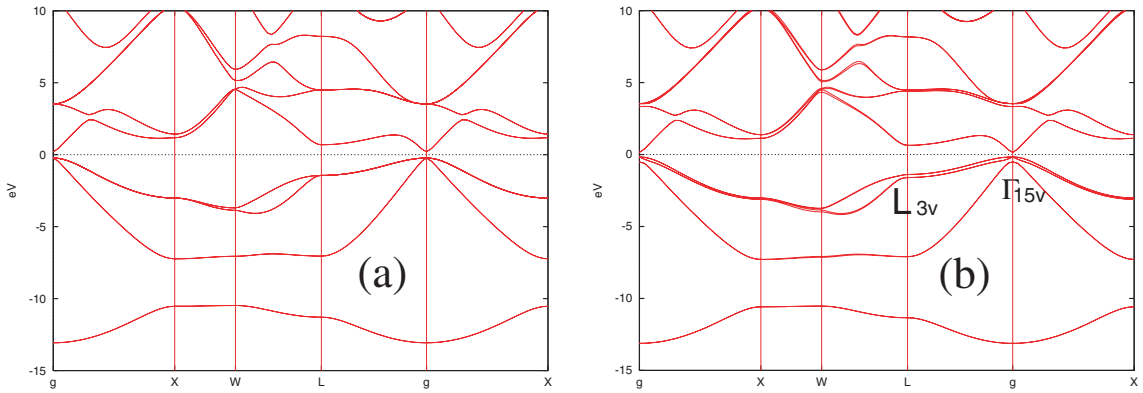



Figure 33: Band structures of a bulk GaAs calculated by the non-collinear DFT (a) without and (b) with the spin-orbit coupling. In these calculations, Ga7.0-s2p2d2 and As7.0-s2p2d2 were used as a basis set, and Ga_CA19.vps and As_CA19.vps were used for pseudopotentials, which are stored in the database Ver. 2019. For the exchange-correlation terms, LDA was used. We used 12× 12× 12 and 140 (Ryd) for scf.Kgrid and scf.energycutoff, respectively. Also the experimental value (5.65Å) was used for the lattice constant. The input file is 'GaAs.dat' in the directory 'work'.

Table 5: Calculated spin-orbit splittings (eV) at the $\Gamma_{15v}$ and the $L_{3v}$ of a bulk GaAs. The other theoretical values (LMTO: Ref. [113], PP: Ref. [114]) and experimental value (Ref.[115]) are also shown for comparison. The calculation conditions are given in the caption of Fig. 33 and the input file is 'GaAs.dat' in the directory 'work'.

| Level | OpenMX | LMTO | PP | Expt. |
|---|---|---|---|---|
| $\Gamma_{15v}$ | 0.344 | 0.351 | 0.35 | 0.34 |
| $L_{3v}$ | 0.213 | 0.213 | 0.22 | |

Then, the spin-orbit coupling can be self-consistently incorporated within the pseudopotential scheme rather than a perturbation scheme. Due to the spin-orbit coupling, $\alpha$ and $\beta$ spin components in the two component spinor can directly interact. In order to determine the absolute spin orientation in the non-collinear DFT calculations, you have to include the spin-orbit coupling, otherwise the spin orientation is not uniquely determined in real space. As an illustration of spin-orbit splitting, we show band structures of a bulk GaAs calculated by the non-collinear DFT without and with spin-orbit coupling in Fig. 33, where the input file is 'GaAs.dat' in the directory 'work'. In Fig. 33(b) we can see that there are spin-orbit splittings in the band dispersion, while no spin-orbit splitting is observed in Fig. 33(a). The spin-orbit splittings at two **k**-points, $\Gamma$ and $L$, are listed together with the other calculations and experimental values in Table 5. We see a good agreement in this table.

### 35.2 Controling of spin-orbit coupling strength

In OpenMX Ver. 3.9, it is possible to contol the spin-orbit coupling strength using a conventional pseudopotential stored in the database Ver. 2019 without generating a special pseudopotential with a larger or smaller spin-orbit coupling. The scaling factors can be specified to each angular momentum quantum number by the following keyword:

```
<scf.SO.factor
  Ga  s 1.0 p 2.0 d 1.0 f 1.0
  As  s 1.0 p 2.0 d 1.0 f 1.0
scf.SO.factor>
```

The first column is the name of species which is defined by the keyword 'Definition.of.Atomic.Species', and followed by the subsequent columns: a symbol for angular momentum quantum number and a scaling factor. The '1.0' corresponds to the spin-orbit coupling strength in the channel with the angular momentum quantum number in the real atom. The set of information needs to be specified up to the f-channel for all the species in your system regardless of a kind of pseudopotentials.

### 35.3 Scalar relativistic treatment

A simple way to incorporate a scalar relativistic treatment is to use scalar relativistic pseudopotentials which can be generated by ADPACK. The another way is to use fully relativistic j-dependent pseudopotentials and to switch off the keyword 'scf.SpinOrbit.Coupling' as follows:

```
scf.SpinOrbit.Coupling      off        # On|Off, default=off
```

Then, the j-dependent pseudopotentials are automatically averaged with a weight of j-degeneracy when they are read by OpenMX, which corresponds to scalar relativistic pseudopotentials. So, once j-dependent pseudopotentials are generated, you can utilize the pseudopotentials for both the fully and scalar relativistic treatments. Thus, we recommend that you make a fully relativistic j-dependent pseudopotential rather than a scalar relativistic pseudopotential, when relativistic effects are taken into account. In fact, the calculation in Fig. 33(a) was performed using the same pseudopotential as in Fig. 33(b) with 'scf.SpinOrbit.Coupling=off'.

# 36 Orbital magnetic moment

The orbital magnetic moment at each atomic site is calculated as default in the non-collinear DFT. Since the orbital magnetic moment appears as a manifestation of spin-orbit coupling (SOC), the calculated values become finite when the SOC is included [118, 119]. As an example, a non-collinear LDA+U (U=5 eV) calculation of iron monoxide bulk is illustrated using an input file 'FeO_NC.dat' in the directory 'work'. As for the LDA+U calculation, see the Section 'LDA+U'. The calculated orbital and spin magnetic moments at the Fe site are listed in Table 4. Also, you can find the orientation of the (decomposed) orbital moment in '*System.Name*.out', where '*System.Name*' means 'System.Name' as follows:

```
***********************************************************
***********************************************************
                    Orbital moments
***********************************************************
***********************************************************

  Total Orbital Moment (muB)   0.000001885   Angles  (Deg) 126.954120326  185.681623854

        Orbital moment (muB)   theta (Deg)  phi (Deg)
   1   Fe   0.76440           131.30039   51.57082
   2   Fe   0.76440            48.69972  231.57071
   3    O   0.00000            40.68612  210.48405
   4    O   0.00000            48.18387  222.72367


 Decomposed Orbital Moments

   1   Fe         Orbital Moment(muB)    Angles (Deg)
          multiple
s           0    0.000000000           90.0000    0.0000
 sum over m      0.000000000           90.0000    0.0000
s           1    0.000000000           90.0000    0.0000
 sum over m      0.000000000           90.0000    0.0000
px          0    0.000055764           42.7669  270.0000
py          0    0.000046795           28.9750  180.0000
pz          0    0.000044132           90.0000  239.0920
 sum over m      0.000120390           47.1503  239.0920
px          1    0.001838092           10.8128  -90.0000
py          1    0.001809013            3.5933  180.0000
pz          1    0.000362989           90.0000  251.7994
 sum over m      0.003683170           11.3678  251.7994
d3z^2-r^2   0    0.043435663           90.0000  224.2874
dx^2-y^2    0    0.066105902           24.3591  229.7056
dxy         0    0.361874370           80.4206   50.6465
```

```
dxz          0    0.397108491          144.2572  -12.7324
dyz          0    0.427070801          138.9995  100.0151
 sum over m       0.776513038          132.4577   51.6984
d3z^2-r^2    1    0.000144144           90.0000  196.4795
dx^2-y^2     1    0.000270422           31.2673  224.0799
dxy          1    0.003006770           85.5910   50.2117
dxz          1    0.002952926          139.3539   -4.1301
dyz          1    0.003222374          134.0513   95.9246
 sum over m       0.006795789          126.2536   52.1993
f5z^2-3r^2   0    0.001903274           90.0000   33.4663
f5xz^2-xr^2  0    0.005186342           14.5594  118.0868
f5yz^2-yr^2  0    0.005258572           17.3323  -35.0807
fzx^2-zy^2   0    0.005477755           29.3372  224.9067
fxyz         0    0.004851020           10.1407  249.0607
fx^3-3*xy^2  0    0.002029489           84.1842  -81.2087
f3yx^2-y^3   0    0.001611593           82.6686  176.3172
 sum over m       0.020307129            9.9551  249.3739
.....
...
```

As shown in Table 6, OpenMX gives a good agreement for both the spin and orbital magnetic moments of a series of 3*d*-transition metal oxides with other calculation results. However, it is noted that the absolute value of orbital magnetic moment seems to be significantly influenced by calculation conditions such as basis functions and on-site 'U' in the LDA+U method, while the spin magnetic moment is relatively insensitive to the calculation conditions, and that a rather rich basis set including polarization functions will be needed for convergent calculations of the orbital magnetic moment.

Table 6: Spin magnetic moment $M_s(\mu_B)$ and orbital magnetic moment $M_o(\mu_B)$ of transition metal oxides, MO (M=Mn, Fe, Co, Ni). In the LDA+U scheme [20], for the first d-orbital of M, the effective U of 3.0 (eV) for Mn, 5.0 (eV) for Fe, Co for 7.0 (eV), and Ni for 7.0 (eV) were used. For the others zero. The local spin moment was calculated by the Voronoi decomposition discussed in the Section 'Voronoi charge' rather than Mulliken charge, since the Mulliken analysis tends to give a larger spin moment in the use of multiple basis functions. The input files are 'MnO_NC.dat', 'FeO_NC.dat', 'CoO_NC.dat', and 'NiO_NC.dat' in the directory 'work'. The other theoretical value [70] and experimental value [70] are also shown for comparison.

| | $M_s$ | | $M_o$ | | |
|---|---|---|---|---|---|
| Compound | OpenMX | Other calc. | OpenMX | Other calc. | Expt. in total |
| MnO | 4.519 | 4.49 | 0.004 | 0.00 | 4.79,4.58 |
| FeO | 3.653 | 3.54 | 0.764 | 1.01 | 3.32 |
| CoO | 2.714 | 2.53 | 1.269 | 1.19 | 3.35,3.8 |
| NiO | 1.687 | 1.53 | 0.247 | 0.27 | 1.77,1.64,1.90 |

# 37 DFT+U methods

OpenMX supports various types of DFT+U methods with respect to the treatment of the occupation number operator, the functional form, and the choice of the double counting term. The functionality is supported for both the collinear and non-collinear calculations. To acknowledge in any publications by using the functionality, the citation of the references [20, 21, 22] would be appreciated. The technical details of the methods and its implementation can be found in Ref. [20, 21, 22] and technical notes [23, 24].

## 37.1 Standard setting

### 37.1.1 Choice of DFT+$U$ scheme; simplified or general

The general forms of DFT+$U$ methods with different definitions of the occupation number operator, the functional form, and the choice of the double counting term are available in OpenMX for both collinear and noncollinear calculations [21, 22] by the following two keywords:

```
scf.Hubbard.U                 on  # on|off, default=off
scf.DFTU.Type                 2   # 1:Simplified(Dudarev)|2:General, default=1
```

`scf.DFTU.Type=1` corresponds to the so-called 'simplified rotationally invariant form' by Dudarev *et al.* [25] as implemented in the previous versions of OpenMX [20], where only $U$ (Hubbard $U$) plays a role. In more general DFT+$U$ schemes, not only $U$ but also $J$ (Hund's coupling $J$) are used as input parameters. Note that the keyword 'scf.SpinPolarization' should be always switched on, meaning that 'on' or 'nc' has to be specified, whenever the DFT+U methods are used.

The occupation number operator [20] is specified by the following keyword:

```
scf.Hubbard.Occupation        dual  # onsite|full|dual, default=dual
```

Among three occupation number operators, only the dual operator satisfies a sum rule that the trace of occupation number matrix gives the total number of electrons. For the details of the operators onsite, full, and dual, see Ref. [20].

The $U$ and $J$ values in eV on each orbital of species defined by

```
<Definition.of.Atomic.Species
  Ni Ni6.0S-s2p2d2f1  Ni_CA13S
  O  O5.0-s2p2d1      O_CA13
Definition.of.Atomic.Species>
```

are specified by

```
<Hubbard.U.values   # eV
  Ni 1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 5.0 2d 0.0 1f 0.0
  O  1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 0.0
Hubbard.U.values>
```

and

```
<Hund.J.values      # eV
  Ni 1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 0.9 2d 0.0 1f 0.0
  O  1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 0.0
Hund.J.values>
```

The beginning of the description must be `<Hubbard.U.values` (`<Hund.J.values`), and the last of the description must be `Hubbard.U.values>` (`Hund.J.values>`) for $U$ ($J$). For all the basis orbitals, you have to give $U$ and $J$ values in eV as in the above format. The '1s' and '2s' mean the first and second $s$-orbital, and the number behind '1s' is the $U$ (or $J$) value for the first $s$-orbital. The same rule is applied to $p$- and $d$-orbitals. When `scf.DFTU.Type=1`, only $U$ values are read and thus users do not need to specify $J$ values.

### 37.1.2 Choice of the double-counting

A double-counting (dc) correction is common for any 'embedding' methods such as DFT+$U$. When using `scf.DFTU.Type=2`, the dc term should be specified by the following keyword:

```
scf.dc.Type                  cFLL    # sFLL|sAMF|cFLL|cAMF, default=sFLL
```

In the above case, the 'cFLL' dc-term is chosen. In the specification of 'sFLL', 'sAMF', 'cFLL', and 'cAMF', 'c' and 's' mean the density functional scheme within charge (spin-unpolarized) density and spin density LDA/GGA, respectively, and 'FLL' and 'AMF' correspond to a fully localized limit (FLL) and around mean-field (AMF) for the treatment of the double-counting term. For the detailed definitions and behaviors of each scheme, please refer to Ref. [21, 22]. Users should keep in mind that when `cFLL` or `cAMF` dc-term is chosen, spin-density exchange-correlation energy of LDA (or GGA) is not taken into account during the SCF loop [21, 22]. `scf.dc.Type=sFLL` corresponds to the form proposed by Liechtenstein *et al.* [26]. Note also that when using a simplified scheme (`scf.DFTU.Type=1`), `sFLL` dc-term is implicit in its functional form.

Users can check what kinds of DFT+$U$ scheme has been specified with the following message before SCF loop begins in the standard output:

```
For scf.DFTU.Type=1,

*******************************************************
                  DFT+U Type and DC
*******************************************************
             scf.DFTU.Type: 1(Simplified)


For scf.DFTU.Type=2 and scf.dc.Type=cFLL,

*******************************************************
                  DFT+U Type and DC
*******************************************************
     scf.DFTU.Type: 2(General)   scf.dc.Type: cFLL
```

As an example of the DFT+$U$ calculation, the density of states for NiO bulk is shown in Fig. 34 for cases with $U = 5$ eV and two different $J$ values (0.5 and 1.0 eV) for $d$-orbitals of Ni. The input files,

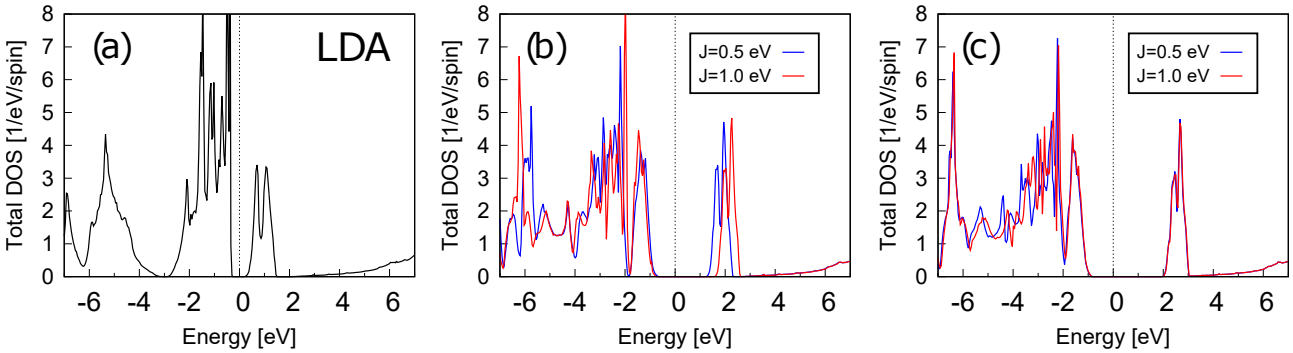



Figure 34: The up-spin density of states of NiO by (a) LDA, (b) DFT+$U$ with 'scf.dc.Type=cFLL', and (c) DFT+$U$ with 'scf.dc.Type=sFLL'. $U$ is fixed to 5 eV and $J = 0.5$ eV for blue lines and 1.0 eV for red lines.

'NiO-cFLL.dat' and 'NiO-sFLL.dat' can be found in the directory 'work'. We can see that the gap increases due to the introduction of $U$ on the $d$-orbitals as well as the different behaviors of varying $J$ depending on the choice `scf.dc.Type`.

The occupation number for each orbital is output to the file '*System.Name*.out' in the same form as that of decomposed Mulliken populations which starts from the title 'Occupation Number in LDA+U', e.g., NiO with 'scf.dc.Type=cFLL' of $U = 5$ eV and $J = 0.5$ eV, as follows:

```
************************************************************
************************************************************
       Occupation Number in LDA+U and Constraint DFT

    Eigenvalues and eigenvectors for a matrix consisting
          of occupation numbers on each site
************************************************************
************************************************************

    1    Ni

     spin= 0

  Sum =  8.708572022602

                    1        2        3        4        5        6        7        8
  Individual     -0.0041   0.0012   0.0012   0.0022   0.0040   0.0040   0.0044   0.0064

  s           0   0.1792 -0.0008 -0.0000  0.0015 -0.0000  0.0003  0.0124 -0.0000
  s           1  -0.9756  0.0052  0.0000  0.0026  0.0000 -0.0041 -0.1251  0.0000
  px          0   0.0006  0.0007 -0.0012 -0.0123  0.0003  0.0006 -0.0033 -0.0000
  py          0   0.0006 -0.0013 -0.0000 -0.0122  0.0000  0.0000 -0.0033  0.0000
  pz          0   0.0006  0.0007  0.0012 -0.0123 -0.0003  0.0006 -0.0033  0.0000
  px          1   0.0091  0.0053 -0.0095 -0.0867  0.0205  0.0152 -0.0206  0.0026
```

```
py          1   0.0093 -0.0116 -0.0000 -0.0870 -0.0000 -0.0207 -0.0236 -0.0000
pz          1   0.0091  0.0052  0.0095 -0.0867 -0.0205  0.0152 -0.0206 -0.0026
d3z^2-r^2   0   0.0002  0.0348  0.0604 -0.0000 -0.0020  0.0012  0.0001 -0.0005
dx^2-y^2    0   0.0004  0.0604 -0.0348 -0.0000  0.0011  0.0020  0.0001  0.0003
dxy         0  -0.0001  0.0007  0.0012  0.0151  0.0367 -0.0218  0.0097 -0.0003
dxz         0  -0.0006 -0.0015 -0.0000  0.0167  0.0000  0.0417  0.0112  0.0000
dyz         0  -0.0001  0.0007 -0.0012  0.0151 -0.0367 -0.0218  0.0097  0.0003
d3z^2-r^2   1  -0.0025 -0.4966 -0.8626 -0.0006  0.0295 -0.0174 -0.0006  0.0056
dx^2-y^2    1  -0.0042 -0.8625  0.4967 -0.0010 -0.0170 -0.0301 -0.0010 -0.0033
dxy         1   0.0136 -0.0160 -0.0276 -0.5343 -0.7016  0.4220 -0.1326  0.0055
dxz         1   0.0225  0.0332  0.0000 -0.5657 -0.0000 -0.7918 -0.1607 -0.0000
dyz         1   0.0136 -0.0161  0.0275 -0.5343  0.7016  0.4219 -0.1325 -0.0055
f5z^2-3r^2  0  -0.0029  0.0032  0.0065 -0.0804 -0.0514  0.0334 -0.0282 -0.0069
f5xz^2-xr^2 0   0.0017 -0.0304 -0.0148  0.0467 -0.0113 -0.0653  0.0174 -0.4673
f5yz^2-yr^2 0   0.0013  0.0057 -0.0294  0.0479  0.0517  0.0341  0.0272  0.4428
fzx^2-zy^2  0  -0.0001 -0.0360  0.0237 -0.0031 -0.0256 -0.0567  0.0001  0.5857
fxyz        0   0.1218 -0.0003 -0.0000  0.2573  0.0000  0.0172 -0.9581  0.0000
fx^3-3*xy^2 0  -0.0023 -0.0195 -0.0197 -0.0655  0.0563 -0.0083 -0.0223 -0.3532
f3yx^2-y^3  0   0.0017  0.0072  0.0228  0.0618 -0.0401  0.0441  0.0352 -0.3430

                  9      10      11      12      13      14      15      16
Individual      0.0116  0.0117  0.0207  0.0207  0.0238  0.0972  0.1112  0.1114

s           0  -0.0003 -0.0000  0.0000 -0.0005 -0.0075 -0.0206 -0.0000 -0.0000
s           1   0.0001  0.0000  0.0000 -0.0013  0.0076  0.0102  0.0000 -0.0000
px          0  -0.0005  0.0006 -0.0024  0.0014  0.0043 -0.0270  0.0291  0.0170
py          0   0.0006  0.0000  0.0000 -0.0027  0.0044 -0.0279 -0.0000 -0.0338
pz          0  -0.0005 -0.0006  0.0024  0.0014  0.0043 -0.0270 -0.0291  0.0171
px          1   0.0229 -0.0402  0.1442 -0.0832 -0.1073  0.5479 -0.6901 -0.4038
py          1  -0.0437  0.0000 -0.0005  0.1632 -0.1127  0.5594  0.0003  0.7916
pz          1   0.0229  0.0402 -0.1437 -0.0841 -0.1073  0.5478  0.6898 -0.4043
d3z^2-r^2   0   0.0053  0.0093  0.0012  0.0006 -0.0001  0.0003 -0.0202  0.0115
dx^2-y^2    0   0.0092 -0.0053 -0.0007  0.0011 -0.0002  0.0006  0.0117  0.0199
dxy         0  -0.0033 -0.0056 -0.0049 -0.0032 -0.0237  0.0916  0.0095 -0.0067
dxz         0   0.0069 -0.0000 -0.0000  0.0054 -0.0236  0.0915  0.0000  0.0102
dyz         0  -0.0033  0.0056  0.0049 -0.0031 -0.0237  0.0916 -0.0095 -0.0067
d3z^2-r^2   1  -0.0241 -0.0404 -0.0230 -0.0138  0.0011 -0.0001  0.0089 -0.0052
dx^2-y^2    1  -0.0418  0.0233  0.0134 -0.0238  0.0018 -0.0002 -0.0051 -0.0090
dxy         1   0.0224  0.0367  0.0645  0.0399  0.1012 -0.0657 -0.0086  0.0058
dxz         1  -0.0489  0.0000  0.0002 -0.0737  0.1010 -0.0652 -0.0000 -0.0096
dyz         1   0.0224 -0.0367 -0.0648  0.0395  0.1011 -0.0657  0.0086  0.0058
f5z^2-3r^2  0   0.0928  0.1648 -0.6676 -0.3997 -0.5498 -0.1226 -0.1505  0.0868
f5xz^2-xr^2 0   0.4854  0.4030 -0.3328  0.3674  0.3359  0.0744 -0.0944 -0.0506
f5yz^2-yr^2 0   0.1112  0.6352  0.1497 -0.4674  0.3502  0.0772 -0.0023  0.1046
```

```
fzx^2-zy^2  0   0.6859 -0.3821 -0.0991  0.1577 -0.0010 -0.0008  0.0028  0.0032
fxyz        0   0.0052  0.0000  0.0000 -0.0109  0.0195  0.0064  0.0000  0.0007
fx^3-3*xy^2 0   0.4934  0.1037  0.5899 -0.2158 -0.4352 -0.0974  0.1173  0.0705
f3yx^2-y^3  0   0.1435 -0.4920 -0.1130 -0.6043  0.4520  0.0997  0.0019  0.1351

                 17      18      19      20      21      22      23      24
Individual      0.2342  0.9866  0.9949  0.9950  1.0070  1.0070  1.0101  1.0101

s           0   0.9835 -0.0030 -0.0001  0.0000 -0.0000  0.0001  0.0003  0.0000
s           1   0.1796 -0.0015 -0.0000  0.0000  0.0000 -0.0000 -0.0000 -0.0000
px          0  -0.0023 -0.5138 -0.3934 -0.6753 -0.0494 -0.0317  0.1133 -0.2016
py          0  -0.0021 -0.5218  0.7787 -0.0000 -0.0000  0.0587 -0.2264 -0.0000
pz          0  -0.0023 -0.5138 -0.3933  0.6753  0.0494 -0.0317  0.1132  0.2017
px          1   0.0092 -0.0625 -0.0195 -0.0329  0.0012  0.0006 -0.0049  0.0083
py          1   0.0095 -0.0631  0.0376 -0.0000  0.0000 -0.0016  0.0097  0.0000
pz          1   0.0092 -0.0625 -0.0195  0.0329 -0.0012  0.0006 -0.0049 -0.0083
.....
...
```

The eigenvalues of the occupation number matrix of each atomic site correspond to the occupation number to each local state given by the eigenvector.

### 37.1.3 Orbital polarization

The DFT+$U$ functional possesses multiple minima in the degree of freedom of the orbital occupation, leading to that the SCF calculation tends to be trapped to some local minimum. To find the ground state with an orbital polarization, a way of enhancing explicitly the orbital polarization is available by the following switch:

```
For collinear cases

<Atoms.SpeciesAndCoordinates   # Unit=AU
  1 Ni 0.0     0.0     0.0     10.0 6.0  on
  2 Ni 3.94955 3.94955 0.0     6.0  10.0 on
  3 O  3.94955 0.0     0.0     3.0  3.0  on
  4 O  3.94955 3.94955 3.94955 3.0  3.0  on
Atoms.SpeciesAndCoordinates>

For noncollinear cases

<Atoms.SpeciesAndCoordinates   # Unit=AU
  1 Ni 0.0     0.0     0.0     10.0 6.0  40.0 10.0 0 on
  2 Ni 3.94955 3.94955 0.0     6.0  10.0 40.0 10.0 0 on
  3 O  3.94955 0.0     0.0     3.0  3.0  10.0 40.0 0 on
  4 O  3.94955 3.94955 3.94955 3.0  3.0  10.0 40.0 0 on
Atoms.SpeciesAndCoordinates>
```

The specification of each column can be found in the section 'Non-collinear DFT'. Since the enhancement treatment for the orbital polarization is performed on each atom, you have to set the switch for all the atoms in the specification of atomic coordinates as given above. The enhancement for the atoms switched on is applied during the first few self-consistent (SC) steps, then no more enhancement are required during the subsequent SC steps. It is also emphasized that the enhancement does not always give the ground state, and that it can work badly in some case (see Ref. [20] for the details). Also, we *do not recommend* turning on orbital polarization when using `scf.dc.Type=cFLL` and `cAMF`.

## 37.2 Additional functionalities

The standard DFT+$U$ calculations can be performed with the keywords mentioned above. In this Section, we present additional functionalities available in OpenMX. These functionalities are available only for `scf.DFTU.Type=2`.

### 37.2.1 Varying the ratio of two Slater integrals ($F^4/F^2$)

The general DFT+$U$ scheme by setting `scf.DFTU.Type=2` requires the generation of Coulomb interaction tensor using two input values, $U$ and $J$. This can be done by the combinations of Slater integrals ($F^0, F^2, F^4, ...$) and Racah-Wigner coefficients assuming atomic sphericity. For $d$-orbitals in the standard DFT+$U$ scheme, $F^4/F^2$ ratio should be specified, and $F^4/F^2 = 0.625$ is the usual choice. This ratio is, however, valid only in the atomic environment and deviates from it in the solid. Users can manually choose the ratio for their own purposes by the following keyword:

```
scf.Slater.Ratio                  0.75    # default=0.625
```

### 37.2.2 Estimation of $J$ and $F^4/F^2$ from input parameter $U$

One can estimate $J$ and $F^4/F^2$ from input $U$ value via Yukawa-type screened Coulomb potential by the following keyword:

```
scf.Yukawa                        on      # default=off
```

By setting `scf.Yukawa=on`, only $U$ values in `<Hubbard.U.values ... Hubbard.U.values>` are read and $J$ values in `<Hund.J.values ... Hund.J.values>` and `scf.Slater.Ratio` are ignored. Instead, $J$ values and $F^4/F^2$ will be automatically generated by estimating the Thomas-Fermi screening length corresponding to the input $U$ values [21, 22]. As an example, for the following orbitals defined by:

```
<Definition.of.Atomic.Species
  Ni Ni6.0S-s2p2d2f1  Ni_CA13S
  O  O5.0-s2p2d1      O_CA13
Definition.of.Atomic.Species>
```

when `scf.Yukawa=on` and $U$ values are set to:

```
<Hubbard.U.values   # eV
  Ni 1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 6.4 2d 3.0 1f 0.0
  O  1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 0.0
Hubbard.U.values>
```

one can see the following message in the standard output before SCF loop begins:

```
*****************************************************
      Calculating Thomas-Fermi screening length
*****************************************************

<species: Ni, angular momentum= 2, multiplicity number= 0>
 TF-screening-length lambda= 1.340787 1/au
 Hubbard U= 6.400000 eV
 Hund J= 1.154943 eV
 Slater F0= 6.399932 eV
 Slater F2= 9.321381 eV
 Slater F4= 6.847820 eV
 F4/F2= 0.734636

<species: Ni, angular momentum= 2, multiplicity number= 1>
 TF-screening-length lambda= 0.271461 1/au
 Hubbard U= 3.000000 eV
 Hund J= 0.380184 eV
 Slater F0= 3.000017 eV
 Slater F2= 2.970544 eV
 Slater F4= 2.352036 eV
 F4/F2= 0.791786
```

These $U$, $J$, and $F^4/F^2$ values will be used for a subsequent DFT+$U$ SCF loop. The input file, 'NiO-Yukawa.dat' can be found in the directory 'work'.

## 38 Constraint DFT for non-collinear spin orientation

To calculate an electronic structure with an arbitrary spin orientation in the non-collinear DFT, OpenMX Ver. 3.9 provides two kinds of constraint functionals which give a penalty unless the difference between the calculated spin orientation and the initial one is zero [13]. The constraint DFT for the non-collinear spin orientation is available by the following keywords:

```
scf.Constraint.NC.Spin       on      # on|on2|off, default=off
scf.Constraint.NC.Spin.v    0.5      # default=0.0(eV)
```

You can introduce the penalty functional by either 'on' or 'on2' for the keyword 'scf.Constraint.NC.Spin'. By 'on', the spin direction is constrained at the initial orientation, but the magnitude of the spin moment can vary so that the total energy can be stabilized. On the other hand, by 'on2', the spin direction and the magnitude of the spin moment are constrained at the initial setting by the keyword 'Atoms.SpeciesAndCoordinates'. The keyword 'scf.Constraint.NC.Spin.v' gives a magnitude which determines the strength of constraint when the constraint is introduced. The constraint is applied on each atom by specifying a flag as follows:

```
<Atoms.SpeciesAndCoordinates
   1   Cr    0.00000    0.00000    0.00000  7.0  5.0 -20.0 0.0  0.0 0.0  1  off
   2   Cr    0.00000    2.00000    0.00000  7.0  5.0  20.0 0.0  0.0 0.0  1  off
Atoms.SpeciesAndCoordinates>
```

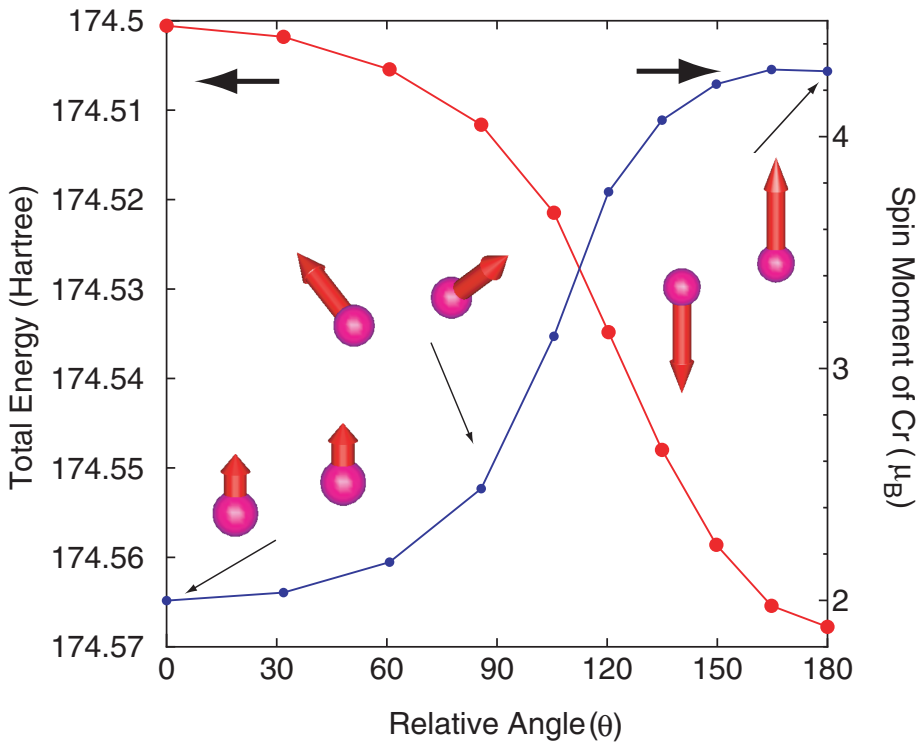



Figure 35: Total energy and magnetic moment of Cr atom for a chromium dimer of which bond length is 2.0 Å. The input file is 'Cr2_CNC.dat' in the directory 'work'.

The '1' in the 12th column means that the constraint is applied, and '0' no constraint. The method constrains only the spin orientation. Therefore, the magnitude of spin can vary. Also the constraint scheme is compatible with the DFT+$U$ calculation explained in the Section 'DFT+$U$'. As an illustration of this method, the dependence of the total energy and magnetic moment in a chromium dimer on the relative angle between two local spins is shown in Fig. 35. You can trace the calculation using an input file 'Cr2_CNC.dat' in the directory 'work'.

Here we provide tips to calculate a smooth energy curve as a function of spin rotational angle using the constraint DFT method. In many cases, local magnetic moments arises from partially filled localized $d$- and/or $f$-orbitals such as $3d$-orbitals. In this case, the occupation of electrons to the localized orbitals may lead to multiple local minima due to the degree of freedom of the occupation. To avoid such a local minima problem, you can start the calculation with a spin orientation along an easy axis. Once the calculation finishes, the restart file is generated. In the next calculation, you may rotate the spin orientation moderately, e.g., 30 degree, compared to the first calculation, and read the restart file which was generated by the first calculation. The same procedure can be performed for all the spin rotational angle that you want to calculate, and at each calculation the restart file generated by the previous step should be read. The step-by-step approach is very effective to avoid the local minima problem.

# 39 Second variational method: Magnetic Anisotropy Energy (MAE)

In the previous section, we introduced a constraint DFT method to control the spin direction. The constraint DFT method enables us to evaluate a magnetic anisotropy energy (MAE) for magnetic systems in a self-consistet manner based on the total energy. However, the computation tends to become very costly, and the calculation tends to be trapped to a local minima due to the degree of freedom of the occupation to degenerate localized states such as $d$-orbitals. A way to bypass these problems is to use a second variational method that a one-shot diagonalization within the non-colliear DFT is performed with spin-orbit coupling (SOI) and the charge density calculated by the collinear DFT as initial guess after getting the SCF charge density within the collinear DFT. Since the variational scheme is based on the Harris functional [14], the perturbation by the spin rotation and the SOI is taken into account only in the band energy. The double counting term does not depend on the spin rotation, while it looks changed in the output of OpenMX, since the energy terms are calculated by the output density rather than the input density (please don't be confused by the output). Using the second variational method, we first calculate a ferromagnetic state within the collinear DFT, resulting in the SCF charge density. Then, the one-shot diagonalization for the Hamiltonian including SOI is performed within the non-colliear DFT using the restart file storing the SCF charge density. The restart file can be read by the following keywords:

```
scf.restart.filename              FePt
scf.restart                       c2n
```

Using the keyword 'scf.restart.filename', the restart file to be read is specified. By 'c2n' for the keyword 'scf.restart' one can port a restart file generated by a collinear DFT calculation to a non-collinear DFT calculation. As an example, the restart file above is generated by an input file 'FePt.dat' available in the directory 'work'. In the second calculation, the spin direction can be specified by the following keywords:

```
scf.Restart.Spin.Angle.Theta      90.0
scf.Restart.Spin.Angle.Phi         0.0
```

The spin direction at all the spatial points is aligned along the direction determined by the two keywords which specify Euler angles. Therefore, it should be noted that the evaluation of the MAE by the second variational method is valid only for ferromagnetic systems. Let us demonstrate how the second variational method works to evaluate the MAE. Figure 36 shows the MAE curves for the spin rotational angle in the $L1_0$ FePt bulk. The result of 'Full SCF' was obtained by the constraint DFT method as explained in the section 38. The input file 'FePt-NC-SCF.dat' is available in the directory 'work'. At each spin rotational angle, the fully self-consistent calculation was performed including the SOI within the non-collinear DFT, where the electronic temperature is 300 K and the k-grid is $17^3$. The MAE of 2.78 meV/f.u. was obtained by 'Full SCF'. Two computational results by the second variational method are also shown in the Fig. 36. Using the restart file generated by the collinear calculation with the input file 'FePt.dat' in the directory 'work', we diagonalized once using an input file 'FePt-NC.dat' available in the directory 'work'. One is the result where the same electronic temperature (300K) and the same k-grid were used as for the 'Full SCF'. It is found that the MAE is 2.89 meV/f.u. The other is the result where the electronic temperatures (the k-grid) for the SCF calculation and the second calculation are 800 ($9^3$) and 300 K ($17^3$), respectively. The MAE

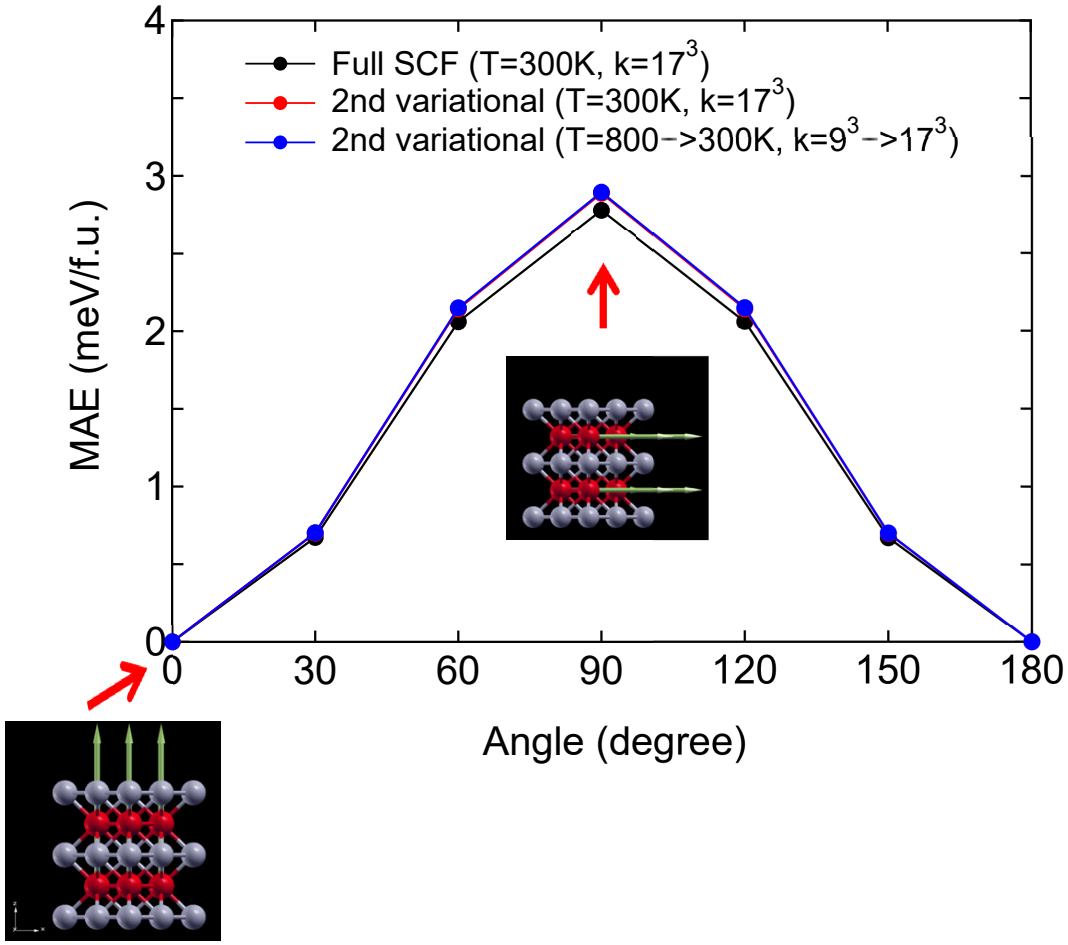



Figure 36: Magnetic anisotropy energy (MAE) curves for the spin rotational angle in the $L1_0$ FePt bulk. The input file used for 'Full SCF' is 'FePt-NC-SCF.dat', and input files 'FePt.dat' and 'FePt-NC.dat' were used for the two calculations by the second variational method. These input files are found in the directory 'work'.

is found to be 2.90 meV/f.u. in the case. Since the two caculations by the second variational method give almost the same result, the fact may allow us to control the electronic temperature and the k-grid for increasing the efficiency and accuracy for both the collinear caculation to generate the SCF charge density and non-collinear calculation to include the SOI. In the second variational method, it is very important to notice that we should look at 'Uele' instead of 'Utot' in the output file. Since the second variational scheme is based on the Harris functional, the MAE should be evaluated by 'Uele', while 'full SCF' relies on the total energy 'Utot'. We see that the difference between the full SCF calculation and the second variational method in evaluating the MAE is about 0.1 meV/f.u. Thus, it might be concluded that the second variational method is an efficient approach for the evaluation of MAE by considering the accuracy and efficiency.

The second variational scheme might be useful for not only the evaluation of MAE, but also calculations of band structure of large-scale systems, which may hamper direct SCF calculations by the non-collinear DFT method, to investigate how the SOI modifies the band structure.

# 40 Zeeman terms

It is possible to apply Zeeman terms to spin and orbital magnetic moments.

## 40.1 Zeeman term for spin magnetic moment

The Zeeman term for spin magnetic moment is available as an interaction with a uniform magnetic field by the following keywords:

```
scf.NC.Zeeman.Spin           on        # on|off, default=off
scf.NC.Mag.Field.Spin       100.0      # default=0.0(Tesla)
```

When you include the Zeeman term for spin magnetic moment, switch on the keyword 'scf.NC.Zeeman.Spin'. The magnitude of the uniform magnetic field can be specified by the keyword 'scf.NC.Mag.Field.Spin' in units of Tesla. Moreover, we extend the scheme as a constraint scheme in which the direction of the magnetic field can be different from each atomic site atom by atom. Then, the direction of magnetic field for spin magnetic moment can be controlled, for example, by the keyword: 'Atoms.SpeciesAndCoordinates'

```
<Atoms.SpeciesAndCoordinates
 1  Sc  0.000  0.000  0.000   6.6 4.4  10.0 50.0  160.0 20.0  1  on
 2  Sc  2.000  0.000  0.000   6.6 4.4  80.0 50.0  160.0 20.0  1  on
Atoms.SpeciesAndCoordinates>
```

The 8th and 9th columns give the Euler angles, $\theta$ and $\phi$, in order to specify the magnetic field for spin magnetic moment. The 12th column is a switch to the constraint. '1' means that the constraint is applied, and '0' no constraint. Since for each atomic site a different direction of the magnetic field can be applied, this scheme provides a way of studying non-collinear spin configuration. It is noted that the keyword 'scf.NC.Zeeman.Spin' and the keyword 'scf.Constraint.NC.Spin' are mutually exclusive. Therefore, when 'scf.NC.Zeeman.Spin' is 'on', the keyword 'scf.Constraint.NC.Spin' must be switched off as follows:

```
scf.Constraint.NC.Spin       off       # on|off, default=off
```

Although the Zeeman term and the constraint scheme for spin orientation can be regarded as ways for controlling the spin orientation, it is noted that the magnitude of spin magnetic moment by the Zeeman term tends to be enhanced unlike the constraint scheme.

## 40.2 Zeeman term for orbital magnetic moment

The Zeeman term for orbital magnetic moment is available as an interaction with a uniform magnetic field by the following keywords:

```
scf.NC.Zeeman.Orbital        on        # on|off, default=off
scf.NC.Mag.Field.Orbital    100.0      # default=0.0(Tesla)
```

When you include the Zeeman term for orbital magnetic moment, switch on the keyword 'scf.NC.Zeeman.Orbital'. The magnitude of the uniform magnetic field can be specified by the keyword 'scf.NC.Mag.Field.Orbital' in units of Tesla. Moreover, we extend the scheme as a constraint scheme in which the direction of the magnetic field can be different from each atomic site atom by atom. Then, the direction of magnetic field for orbital magnetic moment can be controlled, for example, by the keyword: 'Atoms.SpeciesAndCoordinates'

```
<Atoms.SpeciesAndCoordinates
 1  Sc  0.000  0.000  0.000   6.6 4.4  10.0 50.0  160.0 20.0  1  on
 2  Sc  2.000  0.000  0.000   6.6 4.4  80.0 50.0  160.0 20.0  1  on
Atoms.SpeciesAndCoordinates>
```

The 10th and 11th columns give the Euler angles, $\theta$ and $\phi$, in order to specify the magnetic field for orbital magnetic moment. The 12th column is a switch to the constraint. '1' means that the constraint is applied, and '0' no constraint. Since for each atomic site a different direction of the magnetic field can be applied, this scheme provides a way of studying non-collinear orbital configuration. Also, it is noted that the direction of magnetic field for orbital magnetic moment can be different from that for spin moment.

# 41 Macroscopic polarization by Berry's phase

The macroscopic electric polarization of a bulk system can be calculated based on the Berry phase formalism [15]. As an example, let us illustrate a calculation of a Born effective charge of Na in a NaCl bulk via the macroscopic polarization.

**(1) SCF calculation**

First, perform a conventional SCF calculation using an input file 'NaCl.dat' in the directory 'work'. Then, the following keyword 'HS.fileout' should be switched on

```
HS.fileout               on     # on|off, default=off
```

When the calculation is completed normally, then you can find an output file 'nacl.scfout' in the directory 'work'.

**(2) Calculation of macroscopic polarization**

The macroscopic polarization is calculated by a post-processing code 'polB' of which input data is 'nacl.scfout'. In the directory 'source', please compile as follows:

```
% make polB
```

When the compilation is completed normally, then you can find an executable file 'polB' in the directory 'work'. Then, please move to the directory 'work', and perform as follows:

```
 % polB nacl.scfout
or
 % polB nacl.scfout < in > out
```

In the latter case, the file 'in' contains the following ingredients:

```
9 9 9
1 1 1
```

In the former case, you will be interactively asked from the program as follows:

```
*****************************************************************
*****************************************************************
 polB:
 code for calculating the electric polarization of bulk systems
 Copyright (C), 2006-2007, Fumiyuki Ishii and Taisuke Ozaki
 This is free software, and you are welcome to
 redistribute it under the constitution of the GNU-GPL.
*****************************************************************
*****************************************************************

Read the scfout file (nacl.scfout)
 Previous eigenvalue solver = Band
 atomnum                    = 2
```

```
 ChemP                       = -0.156250000000 (Hartree)
 E_Temp                      = 300.000000000000 (K)
 Total_SpinS                 =  0.000000000000 (K)
 Spin treatment              = collinear spin-unpolarized

 r-space primitive vector (Bohr)
  tv1=  0.000000   5.319579   5.319579
  tv2=  5.319579   0.000000   5.319579
  tv3=  5.319579   5.319579   0.000000
 k-space primitive vector (Bohr^-1)
  rtv1= -0.590572   0.590572   0.590572
  rtv2=  0.590572  -0.590572   0.590572
  rtv3=  0.590572   0.590572  -0.590572

  Cell_Volume=301.065992 (Bohr^3)

 Specify the number of grids to discretize reciprocal a-, b-, and c-vectors
 (e.g 2 4 3)
   k1    0.00000    0.11111    0.22222    0.33333    0.44444  ...
   k2    0.00000    0.11111    0.22222    0.33333    0.44444  ...
   k3    0.00000    0.11111    0.22222    0.33333    0.44444  ...

 Specify the direction of polarization as reciprocal a-, b-, and c-vectors
 (e.g 0 0 1 ) 1 1 1
```

Then, the calculation will start like this:

```
calculating the polarization along the a-axis ....
The number of strings for Berry phase : AB mesh=81

  calculating the polarization along the a-axis ....   1/  82
  calculating the polarization along the a-axis ....   2/  82
  .....
  ...

*********************************************************
              Electric dipole  (Debye) : Berry phase
*********************************************************

 Absolute dipole moment      163.93373639

                Background        Core              Electron          Total

 Dx           -0.00000000       94.64718996       -0.00000338       94.64718658
```

```
 Dy          -0.00000000       94.64718996       -0.00000283       94.64718713
 Dz          -0.00000000       94.64718996       -0.00000317       94.64718679


*****************************************************************
             Electric polarization (muC/cm^2) : Berry phase
*****************************************************************

              Background        Core              Electron          Total

 Px          -0.00000000      707.66166752       -0.00002529      707.66164223
 Py          -0.00000000      707.66166752       -0.00002118      707.66164633
 Pz          -0.00000000      707.66166752       -0.00002371      707.66164381


Elapsed time = 77.772559 (s) for myid=  0
```

Since the Born effective charge $Z^*_{\alpha\beta}$ is defined by a tensor:

$$Z^*_{\alpha\beta} = \frac{V_c}{|e|}\frac{\Delta P_\alpha}{\Delta u_\beta}$$

where $V_c$ is the volume of the unit cell, $e$ the elementary charge, $\Delta u_\beta$ displacement along $\beta$-coordinate, $\Delta P_\alpha$ the change of macroscopic polarization along $\alpha$-coordinate, therefore we will perform the above procedures (1) and (2) at least two or three times by varying the $x$, $y$, or $z$-coordinate of Na atom. Then, for example, we have along $x$-coordinates

```
Px = 94.39497736 (Debye/unit cell) at x= -0.05 (Ang)
Px = 94.64718658 (Debye/unit cell) at x=  0.0  (Ang)
Px = 94.89939513 (Debye/unit cell) at x=  0.05 (Ang)
```

Thus,

$$\begin{aligned} Z^*_{xx} &= \frac{(94.89939513 - 94.39497736)/(2.54174776)}{0.1/0.529177} \\ &= 1.050 \end{aligned}$$

Table 7: Calculated Born effective charge of Na in a NaCl bulk. The input file is 'NaCl.dat' in the directory 'work'. Another theoretical value (FD: Ref. [116]) and experimental value (Ref. [117]) are also shown for comparison.

| | OpenMX | FD | Expt. |
|---|---|---|---|
| $Z^*$ | 1.05 | 1.09 | 1.12 |

Note that in the NaCl bulk the off-diagonal terms in the tensor of Born charge are zero, and $Z^*_{xx} = Z^*_{yy} = Z^*_{zz}$. In Table 7 we see that the calculated value is in good agreement with the other calculation [116] and an experimental result [117]. The calculation of macroscopic polarization is supported for both the collinear and non-collinear DFT. It is also noted that the code 'polB' has been parallelized for large-scale systems where the number of processors can exceed the number of atoms in the system.

# 42 Exchange coupling parameter

## 42.1 General

The 'jx', a post-processing code for OpenMX, provides a way to calculate exchange coupling parameters $J_{ij}$ between two localized spins based on Green's function representation of Liechtenstein formula [17]. In the standard distribution of OpenMX Ver. 3.9, the evaluation is supported for only the collinear calculations of cluster and bulk systems. To acknowledge in any publications by using the functionality, the citation of the references [18, 19] would be appreciated.

The program provides three ways to calculate $J_{ij}$ as explained below.

**For cluster systems**, the program computes exchange coupling constants $J_{ij}$ between atomic sites $i$ and $j$ from the following formula:

$$\begin{aligned} J_{ij} &= \frac{1}{4}\sum_{n,n'}\frac{-f_{n\uparrow}+f_{n'\downarrow}}{\varepsilon_{n\uparrow}-\varepsilon_{n'\downarrow}} \\ &\quad\times\sum_{\mu,\nu\in i}\sum_{\mu',\nu'\in j} C_{j\mu',n\uparrow}C^*_{i\nu,n\uparrow}[\hat{P}_i]_{\nu\mu}C_{i\mu,n'\downarrow}C^*_{j\nu',n'\downarrow}[\hat{P}_j]_{\nu'\mu'} \end{aligned} \tag{3}$$

$$\hat{P}_i \equiv \hat{H}_{i\uparrow}-\hat{H}_{i\downarrow}, \tag{4}$$

where $\varepsilon_{n\sigma}$ and $\mathbf{C}_{n\sigma}$ represent the corresponding eigenvalue and eigenvector, respectively, indexed by $n$ and $\sigma$ for the Kohn–Sham equation at the wave number $\mathbf{k}$; and $[\hat{P}_i]_{\nu\mu}$ and $[\hat{P}_j]_{\nu'\mu'}$ represent the partial matrices of the potential difference operator at the sites $i$ and $j$, respectively.

**For bulk systems**, the program computes exchange coupling constants $J_{i\mathbf{0},j\mathbf{R}}$ between individual sites $i$ and $j$ located at cells $\mathbf{0}$ and $\mathbf{R}$, respectively, from the following formula:

$$J_{i\mathbf{0},j\mathbf{R}} = \frac{1}{2}\sum_{p=1}^{N_{\rm P}}\tilde{R}_p\sum_{\mu,\nu\in i}\sum_{\mu',\nu'\in j}\mathrm{Re}\left\{[\hat{P}_i]_{\nu\mu}G^+_{i\mu,j\nu'}(\downarrow,\tilde{z}_p,\mathbf{R})[\hat{P}_j]_{\nu'\mu'}G^+_{j\mu',i\nu}(\uparrow,\tilde{z}_p,-\mathbf{R})\right\}. \tag{5}$$

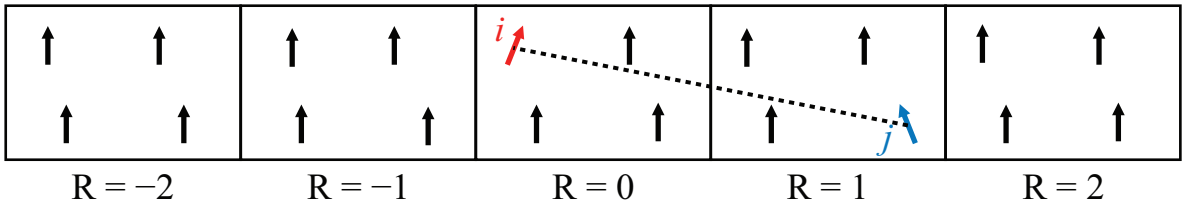
where $\tilde{z}_p$ and $\tilde{R}_p$ are the positive poles and corresponding residues of approximated Fermi functions, $i$ and $j$ are atom indices in a unit cell, $\mathbf{R}$ is a cell index of the second atom, The Green's functions $G^+_{j\mu',i\nu}(\uparrow,\varepsilon,-\mathbf{R})$ and $G^+_{i\mu,j\nu'}(\downarrow,\varepsilon,\mathbf{R})$ are defined as the following equations:

$$G^+_{j\mu',i\nu}(\uparrow,\varepsilon,-\mathbf{R}) \equiv \int d^3\left(\frac{ka}{2\pi}\right)e^{i\mathbf{k}\cdot\mathbf{R}}\sum_n\frac{C_{j\mu',n\uparrow}(\mathbf{k})C_{i\mu,n\uparrow}(\mathbf{k})}{\varepsilon+i\eta-\varepsilon_{n\uparrow}(\mathbf{k})} \tag{6}$$

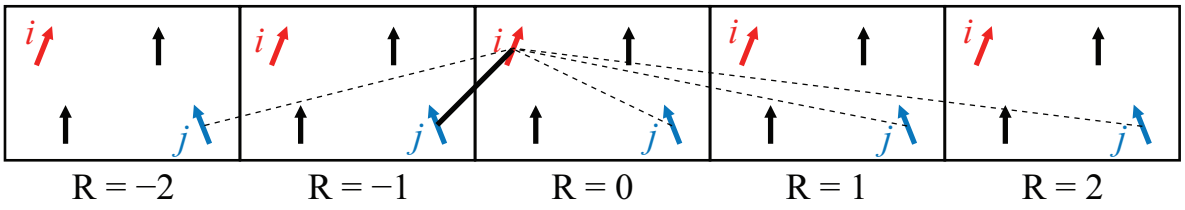
$$G^+_{i\mu,j\nu'}(\downarrow,\varepsilon,\mathbf{R}) \equiv \int d^3\left(\frac{ka}{2\pi}\right)e^{-i\mathbf{k}\cdot\mathbf{R}}\sum_{n'}\frac{C_{i\mu,n'\downarrow}(\mathbf{k})C_{j\nu',n'\downarrow}(\mathbf{k})}{\varepsilon+i\eta-\varepsilon_{n'\downarrow}(\mathbf{k})}. \tag{7}$$

Alternatively, it is also possible to compute the summation of exchange coupling $J_{ij}$ over $\mathbf{R}$ by the following formula:

$$\begin{aligned} J_{ij} &= \sum_{\mathbf{R}}J_{i\mathbf{0},j\mathbf{R}} \\ &= \frac{1}{4}\int d^3\left(\frac{ka}{2\pi}\right)\sum_{n,n'}\frac{-f_{n\uparrow}(\mathbf{k})+f_{n'\downarrow}(\mathbf{k})}{\varepsilon_{n\uparrow}(\mathbf{k})-\varepsilon_{n'\downarrow}(\mathbf{k})} \\ &\quad\times\sum_{\mu,\nu\in i}\sum_{\mu',\nu'\in j}C_{j\mu',n\uparrow}(\mathbf{k})C^*_{i\nu,n\uparrow}(\mathbf{k})[\hat{P}_i]_{\nu\mu}C_{i\mu,n'\downarrow}(\mathbf{k})C^*_{j\nu',n'\downarrow}(\mathbf{k})[\hat{P}_j]_{\nu'\mu'}. \end{aligned} \tag{8}$$




(a) Interaction between individual sites *i* and *j* located at specific cells.




(b) Interaction between periodic images of sites *i* and *j*.

Figure 37: Schematic of exchange coupling constant between individual sites and that between periodic images.

For the details of Eqs. (3), (5), and (8), see Ref. [18, 19].

For users, it might be helpful to explain the difference between Eqs. (5) and (8) by schematics shown in Fig. 37. Figure 37 (a) shows a schematic of interaction between individual sites, which corresponds to Eq. (5). Figure 37 (b) shows a schematic of interaction between periodic images, which corresponds to Eq. (8). When executing, this option can be specified by `Flag.PeriodicSum`, as explained in the latter subsection.

### 42.2 Compilation of jx

To compile 'jx' which is a post-processing code calculating $J_{ij}$ from the OpenMX output, please move to the directory 'source', and type as follows:

```
% make jx
```

When the compilation is completed normally, then you can find an executable file 'jx' in the directory 'work'.

### 42.3 OpenMX calculation to generate jx input

In advance of the $J_{ij}$ calculation, it is necessary to perform a collinear DFT calculation to generate a scfout file. To generate the scfout file, you can add the following keyword 'HS.fileout' in the OpenMX input file as follows:

```
HS.fileout                    on       # on|off, default=off
```

Note that it might be necessary to take a relatively small number of orbitals in the OpenMX calculation for the jx execution, because the pseudo-atomic orbitals have to be localized well in the current implementation. For example, s2p2d2 for Fe and s2p1d1 for Nd should be optimal setting for the atomic orbital specification.

### 42.4 Preparation of config file for jx

For the execution of 'jx' in OpenMX 3.9 implementation, it is necessary to specify a couple of parameters in a configuration file, which should be written in the same format to that of OpenMX input files.

An example of 'jx.config' is shown below:

```
Flag.PeriodicSum           off       # default - off
Num.Poles                  60
Num.Kgrid                  27 27 27
Num.ij.pairs               236
Bunch.ij.pairs             236
<ijpairs.cellid
  1 1 -2 -2 -2
  1 1 -2 -2 -1
  1 1 -2 -2 0
   ...
   ...
  2 2 0 -1 2
  2 2 0 0 -2
  2 2 0 0 -1
ijpairs.cellid>
```

Each keyword is explained below:

```
Flag.PeriodicSum           off       # default - off
```

This flag determines how you calculate the exchange couplings. When 'on' for periodic systems, the program derives the $J_{ij}$ based on Eq. (8). When 'off' for periodic systems, the program derives the $J_{i\mathbf{0},j\mathbf{R}}$ based on Eq. (5). This flag has no role in cluster calculations. For cluster calculations, the program calculates $J_{ij}$ by Eq. (3).

```
Num.Poles                  60
```

This keyword specifies the number of poles $N_{\rm P}$ for the finite pole approximation of Fermi function [74], appearing in Eq. (5). The computation becomes more accurate as increasing the number of poles at the cost of execution time. The most appropriate value depends on the system, and for bcc-Fe a proper value is evaluated as about 60 to attain the accuracy of 0.05 meV. For the electronic temperature of 300 K, the 60 poles might be enough in most cases. However, you can take larger value so as to attain further computational accuracy, because the computational times to construct each exchange coupling were proven much smaller than that of eigenvalue calculation.

```
Num.Kgrid                  27 27 27
```

This keyword specifies the number of $k$-grids. These values should be same or a bit larger than those in OpenMX calculation. No role in cluster calculations.

```
Num.ij.pairs                236        # NOTE: Number of ij pairs.
```

This keyword specifies how many exchange coupling constants to be calculated. The value must be same as the number of rows between the keywords `<ijpairs.cellid` and `ijpairs.cellid>`.

```
<ijpairs.cellid
  1 1 -2 -2 -2
  1 1 -2 -2 -1
  1 1 -2 -2 0
   ...
   ...
  2 2 0 -1 2
  2 2 0 0 -2
  2 2 0 0 -1
ijpairs.cellid>
```

This field specifies the sites $i$, $j$, and cell vector $\mathbf{R} = l_1\mathbf{a}_1 + l_2\mathbf{a}_2 + l_3\mathbf{a}_3$ of $J_{i\mathbf{0},j\mathbf{R}}$, where $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$ are unit lattice vectors in the OpenMX input. The data order of this field is as follows: $i$ $j$ $l_1$ $l_2$ $l_3$. In the case of cluster or periodic image calculations, you should use alternative field as follows:

```
<ijpairs.nocellid
  1 1
  1 2
ijpairs.nocellid>
```

because they do not use cell vectors in $J_{ij}$ calculations.

```
Bunch.ij.pairs          236        # default=Num.ij.pairs
```

This is an optional keyword to use when the memory consumption is too large at the default setting. This value should be same to or smaller than `Num.ij.pairs`. The smaller Bunch.ij.pairs results in smaller memory consumption with larger calculation time.

### 42.5 Execution of jx and MPI parallelization

To execute the program 'jx', it is necessary to type the command in the following orders.

```
% ./jx fe2.scfout jx.config > jx.log
```

The first argument should be a `scfout` file generated by OpenMX. The second argument is the configuration file explained above.

Also, 'jx' supports the MPI parallelization for periodic calculations.

```
% mpirun -np 2 ./jx fe2.scfout jx.config > jx.log
```

Note that MPI parallelization of 'jx' is supported only for the eigenvalue solver 'band' and not for 'cluster'. We also have to note that the parallelization is done only for the $k$ vector, and when the parallelization number becomes larger than the total number of $k$-points it would result in unnecessary consumption of computational resources.

## 42.6 Examples

As an example of the **cluster case**, using an input file 'Fe_Cluster_jx.dat' which is stored in the directory 'work', you can first perform a conventional calculation using 'openmx' as

```
% mpirun -np 2 ./openmx Fe_Cluster_jx.dat
```

The input file 'Fe_Cluster_jx.dat' is for the SCF calculation of a iron dimer. After finishing the calculation normally, you can obtain a scfout file 'Fe_Cluster_jx.scfout'. Then the calculation by 'jx' is performed as

```
% ./jx Fe_Cluster_jx.scfout jx_cluster.config
```

where 'jx_cluster.config' is also available in the directory 'work'. Then, you may see the following message on your screen.

```
*******************************************************************
*******************************************************************
 jx: code for calculating an effective exchange coupling constant J
 Copyright (C), 2003, Myung Joon Han, Jaejun Yu, and Taisuke Ozaki
                2019, Asako Terasawa and Taisuke Ozaki
 This is free software, and you are welcome to
 redistribute it under the constitution of the GNU-GPL.
*******************************************************************
*******************************************************************

Read the scfout file (Fe_Cluster_jx.scfout)
***
The file format of the SCFOUT file:  3
And it supports the following functions:
- jx
- polB
- kSpin
- Z2FH
- calB
***

 Previous eigenvalue solver = Cluster
 atomnum                    = 2
 ChemP                      = -0.089740215968 (Hartree)
 E_Temp                     = 300.000000000000 (K)

Evaluation of J based on cluster calculation

    i     j            J [meV]            J [mRy]
-------------------------------------------------
    1     1   1591.520791120630    116.974621661729
```

```
      1     2    106.511867492210      7.828477938772
      2     2   1591.520746009061    116.974618346089

   Elapsed time = 0.036225 (s)
```

The exchange coupling constant $J_{12}$ between atoms 1 and 2 is calculated to be 106.5 meV.

As an example of the **bulk case**, you can perform a conventional calculation using 'openmx' and an input file 'Fe_Bulk_jx.dat' available in the directory 'work' as

```
% mpirun -np 28 ./openmx Fe_Bulk_jx.dat
```

The input file 'Fe_Bulk_jx.dat' is for the SCF calculation of a BCC iron. After finishing the calculation normally, you can obtain a scfout file 'Fe_Bulk_jx.scfout'. Then the calculation by 'jx' is performed as

```
% mpirun -np 112 ./jx Fe_Bulk_jx.scfout Fe_Bulk_jx.config | tee jx.log
```

where 'Fe_Bulk_jx.config' is available in the directory 'work'. Then, you may see the following message on your screen.

```
*******************************************************************
*******************************************************************
 jx: code for calculating an effective exchange coupling constant J
 Copyright (C), 2003, Myung Joon Han, Jaejun Yu, and Taisuke Ozaki
                2019, Asako Terasawa and Taisuke Ozaki
 This is free software, and you are welcome to
 redistribute it under the constitution of the GNU-GPL.
*******************************************************************
*******************************************************************

Read the scfout file (Fe_Bulk_jx.scfout)
***
The file format of the SCFOUT file:  3
And it supports the following functions:
- jx
- polB
- kSpin
- Z2FH
- calB
***

 Previous eigenvalue solver = Band
 atomnum                    = 2
 ChemP                      = -0.205912787451 (Hartree)
 E_Temp                     = 300.000000000000 (K)

 Jij calculation for a periodic structure
   Number of k-grids: 27 27 27
```

```
   flag_periodic_sum = 0: coupling between site i at cell 0 and site j at cell R
     Number of poles of Fermi-Dirac continued fraction (PRB.75.035123): 60

    i     j    c1    c2    c3           J [meV]           J [mRy]   time_eig [s] ...
------------------------------------------------------------------------------------ ...
    1     1    -2    -2    -2   -0.845809571401   -0.062165857439        0.51534 ...
    1     1    -2    -2    -1    0.274300677331    0.020160728111        0.00000 ...
    1     1    -2    -2     0    0.036006012552    0.002646393135        0.00000 ...
    1     1    -2    -2     1    0.274300705154    0.020160730156        0.00000 ...
    1     1    -2    -2     2   -0.845809596417   -0.062165859278        0.00000 ...
    1     1    -2    -1    -2    0.274300737539    0.020160732536        0.00000 ...
    1     1    -2    -1    -1   -0.206315672897   -0.015163922403        0.00000 ...
    1     1    -2    -1     0    0.149714301525    0.011003798302        0.00000 ...
    1     1    -2    -1     1   -0.206315540488   -0.015163912672        0.00000 ...
    1     1    -2    -1     2    0.274300804604    0.020160737465        0.00000 ...
    ...
    ..
    2     2     0    -1     2    0.149714016159    0.011003777328        0.00000 ...
    2     2     0     0    -2    0.401809366424    0.029532443987        0.00000 ...
    2     2     0     0    -1   11.452192349598    0.841720620155        0.00000 ...

  Elapsed time = 340.817975 (s)
```

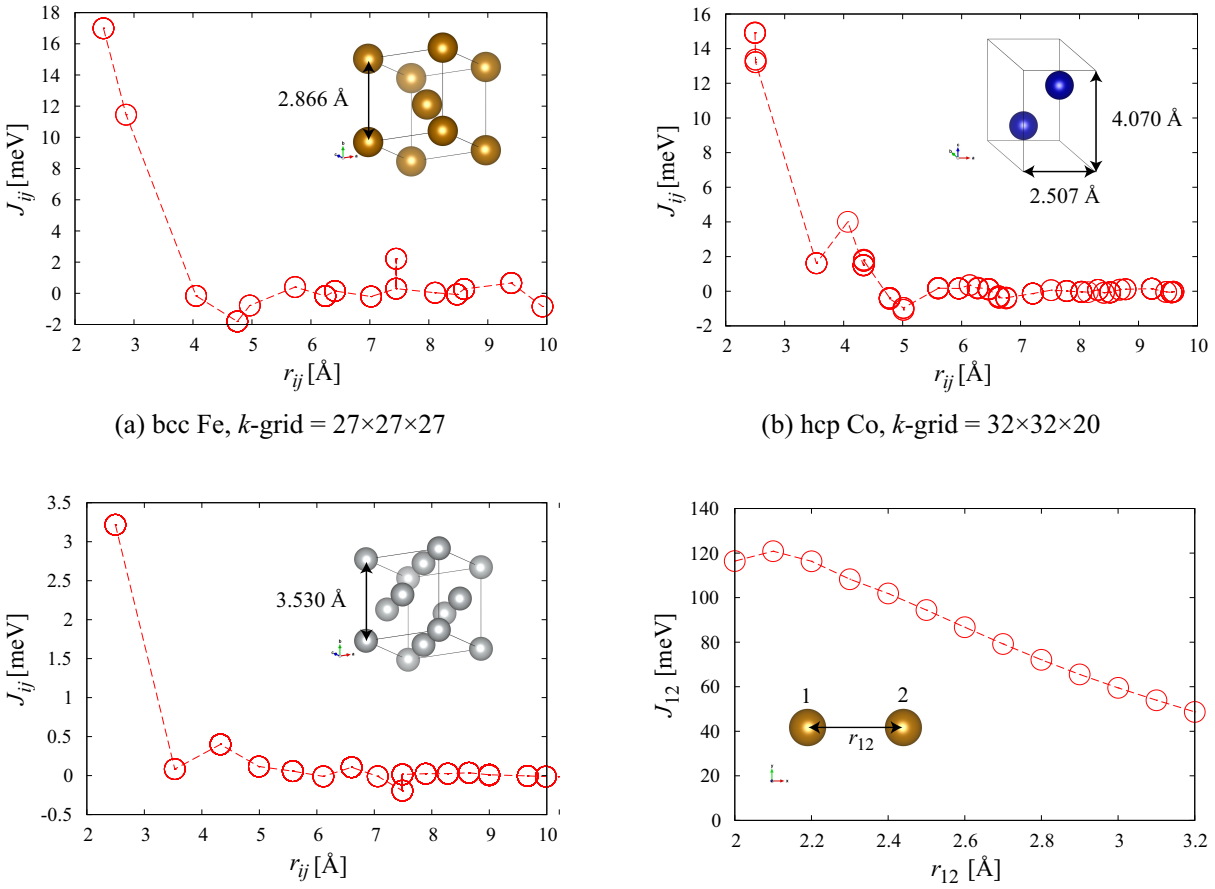
In Fig. 38 (a), the obtained exchange coupling constant $J$ for the bcc Fe is plotted as a function of distance as well as cases of (b) hcp Co, (c) fcc Ni, and (d) Fe dimer. In all the calculations, basis sets of A6.0H-s2p2d2 (A=Fe, Ni, Co), exchange correlation functions of GGA-PBE, and ferromagnetic spin configurations are adopted. The input files used for the calculations are Fe_Bulk_jx.(dat,config), Co_Bulk_jx.(dat,config), Ni_Bulk_jx.(dat,config), and Fe_Cluster_jx.(dat,config) which are all available in the directory 'work'. The details of the calculations can be found in Ref. [19].

It is also possible to calculate the Curie temperature of periodic systems from calculated exchange coupling constants and the mean field approximation. That is, the Curie temperature $T_{\rm C}$ of general periodic system can be obtained as the maximum eigenvalue of the following eigenvalue equation:

$$T\langle\vec{s}_i\rangle_z = \frac{2}{3k_{\rm B}}\sum_j \tilde{J}_{ij}\langle\vec{s}_j\rangle_z \tag{9}$$

$$\tilde{J}_{ij} \equiv J_{ij} - J_{i\mathbf{0},j\mathbf{0}}\delta_{ij}. \tag{10}$$

Here, the definitions of $J_{ij}$ and $J_{i\mathbf{0},j\mathbf{0}}$ are found in Eqs. (5) and (8), respectively. Table 8 shows the calculated Curie temperature by Eq. (9) for the bcc Fe, hcp Co, and fcc Ni together with the experimental values.




(a) bcc Fe, $k$-grid = 27×27×27

(b) hcp Co, $k$-grid = 32×32×20

(c) fcc Ni, $k$-grid = 24×24×24

(d) $Fe_2$ cluster

Figure 38: Calculated exchange coupling constants $J$ for (a) bcc Fe, (b) hcp Co, (c) fcc Ni, and (d) Fe dimer as a function of distance. The input files used for the calculations are Fe_Bulk_jx.(dat,config), Co_Bulk_jx.(dat,config), Ni_Bulk_jx.(dat,config), and Fe_Cluster_jx.(dat,config) which are all available in the directory 'work'.

| | $T_C$ [K] | |
|---|---|---|
| System | calculated | experimental |
| bcc Fe | 1321 | 1040 |
| hcp Co | 1640 | 1131 |
| fcc Ni | 445 | 627 |

Table 8: The calculated Curie temperature from exchange coupling constants and mean field approximation for bcc Fe, hcp Co and fcc Ni together with their experimental values.

# 43 Electric transport calculations

## 43.1 General

Electronic transport properties of molecules, nano-wires, and bulks such as superlattice structures can be calculated based on a non-equilibrium Green function (NEGF) method within the collinear and non-collinear DFT methods. The features and capabilities are listed below:

- SCF calculation of system with two leads under zero and finite bias voltage
- SCF calculation under gate bias voltage
- Compatible with the DFT+$U$ method
- Spin-dependent transmission and current
- **k**-resolved transmission and current along perpendicular to the current axis
- Calculation of current-voltage curve
- Accurate and efficient contour integration scheme
- Interpolation of the effect by the bias voltage
- Quick calculation for periodic systems under zero bias
- The eigen-channel analysis [141]
- The real-space charge and spin current [139]

The details of the implementation can be found in Ref. [73]. First the usage of the functionalities for the collinear case is explained in the following subsections. After that, the non-collinear case will be discussed.

**System we consider**

In the current implementation of OpenMX Ver. 3.9, a system shown in Fig. 39(a) is treated by the NEGF method. The system consists of a central region connected with infinite left and right leads, and the two dimensional periodicity spreads over the **bc**-plane. Considering the two dimensional periodicity, the system can be cast into an one-dimensional problem depending on the Bloch wave vector **k** shown in Fig. 39(b). Also, the Green function of the region $C(\equiv L_0|C_0|R_0)$ is self-consistently determined in order to take account of relaxation of electronic structure around the interface between the central region $C_0$ and the region $L_0(R_0)$. It should be noted that the electronic transport is assumed to be along the **a**-axis in the current implementation. Thus, users have to keep in mind the specification when the geometrical structure is constructed. See also the subsection 'Step 1: The calculations for leads'.

**Computational flow**

The NEGF calculation is performed by the following three steps:

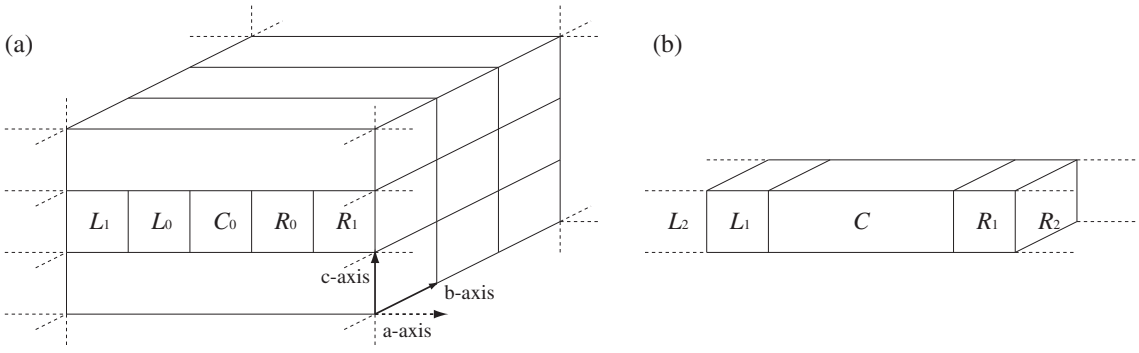



Figure 39: (a) Configuration of the system, treated by the NEGF method, with the infinite left and right leads along the **a**-axis under a two dimensional periodic boundary condition on the **bc**-plane. (b) One dimensional system compacted from the configuration of (a) by considering the periodicity on the **bc**-plane, where the region $C$ is an extended central region consisting of $C_0$, $L_0$, and $R_0$.

**Step 1 → Step 2 → Step 3**

Each step consists of

- **Step 1**

  The band structure calculations are performed for the left and right leads using a program code 'openmx'. The calculated results will be used to represent the Hamiltonian of the leads in the NEGF calculation of the step 2.

- **Step 2**

  The NEGF calculation is performed for the structure shown in Fig. 39 under zero or a finite bias voltage using a program code 'openmx', where the result in the step 1 is used for the construction of the leads.

- **Step 3**

  By making use of the result of the step 2, the transmission, charge/spin current density, and the eigenchannel are calculated by a program code 'openmx'.

**An example: carbon chain**

As a first trial, let us illustrate the three steps by employing a carbon chain.

**Step 1**

```
%./openmx Lead-Chain.dat | tee lead-chain.std
```

A file 'negf-chain.hks' is generated by the step 1.

**Step 2**

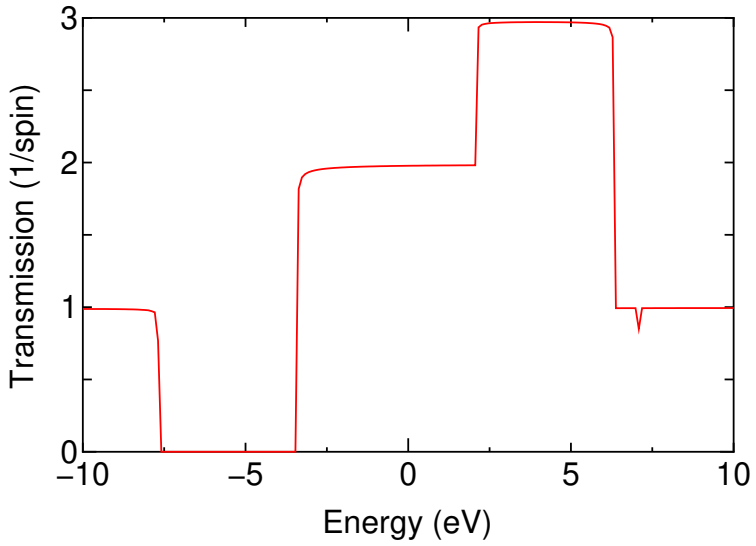



Figure 40: Transmission of a carbon chain as a function of energy. The origin of energy is set to the chemical potential of the left lead.

```
%./openmx NEGF-Chain.dat | tee negf-chain.std
```

A file 'negf-chain.tranb' is generated by the step 2.

**Step 3**

`openmx` starts the step 3 immediately after it finishes the step 2. If we perform separately the step 2 and the step 3, we run `openmx` as follows:

```
%./openmx NEGF-Chain.dat | tee negf-chain.std
```

In the step 3, `openmx` reads a file 'negf-chain.tranb' and, calculates the transmission, current, and eigen channels. After the calculation, the following files:

```
negf-chain.conductance
negf-chain.current
negf-chain.tran0_0
negf-chain.tranec0_0_0_0_i.cube
negf-chain.tranec0_0_0_0_r.cube
negf-chain.tranec0_0_0_1_i.cube
negf-chain.tranec0_0_0_1_r.cube
negf-chain.tranec0_0_0_2_i.cube
negf-chain.tranec0_0_0_2_r.cube
negf-chain.tranec0_0_0_3_i.cube
negf-chain.tranec0_0_0_3_r.cube
negf-chain.tranec0_0_0_4_i.cube
negf-chain.tranec0_0_0_4_r.cube
negf-chain.traneval0_0_0
negf-chain.tranevec0_0_0
```

are generated by the step 3.

The calculations can be traced by using the input files stored in a directory of 'work/negf_example'. By plotting the sixth column in 'negf-chain.tran0_0' as a function of the fourth column, you can see a transmission curve as shown in Fig. 40.

## 43.2 Step 1: The calculations for leads

The calculation of the step 1 is the conventional band structure calculation to construct information of the lead except for adding the following two keywords 'NEGF.output_hks' and 'NEGF.filename.hks':

```
NEGF.output_hks    on
NEGF.filename.hks  lead-chain.hks
```

The calculated results such as Hamiltonian matrix elements, charge distribution, and difference Hartree potential are stored in a file specified by the keyword 'NEGF.filename.hks'. In this case, a file 'lead-chain.hks' is generated. The file '*NEGF.filename.hks*' is used in the calculation of the step 2. Since the electronic transport is assumed to be along the **a**-axis in the current implementation, you have to set the **a**-axis for the direction of electronic transport in the band structure calculation. However, you do not need rotate your structure. All you have to do is to change the specification of the lattice vectors. For example, if you want to specify a vector (0.0, 0.0, 10.0) as the **a**-axis in the following lattice vectors:

```
<Atoms.UnitVectors
  3.0  0.0  0.0
  0.0  3.0  0.0
  0.0  0.0 10.0
Atoms.UnitVectors>
```

you only have to specify as follows:

```
<Atoms.UnitVectors
  0.0  0.0 10.0
  3.0  0.0  0.0
  0.0  3.0  0.0
Atoms.UnitVectors>
```

Then, the direction of (0.0, 0.0, 10.0) becomes the direction of electronic transport. As shown in the above example, when you change the order of the lattice vectors, please make sure that the keyword 'scf.Kgrid' has to be changed as well.

In the calculation of the step 2, the semi-infiniteness of the leads is taken into account by using the surface Green function which allows us to treat the semi-infiniteness without introducing any discretization. Thus, it would be better to use a large number of **k**-points along the **a**-axis to keep the consistency between the steps 1 and 2 with respect to treatment of the semi-infiniteness of the **a**-axis. Also it is noted that the number of **k**-points for the **bc**-plane should be consistent in the steps 1 and 2.

## 43.3 Step 2: The NEGF calculation

### A. Setting up Lead|Device|Lead

You can set up the regions $L_0$, $C_0$, and $R_0$ in the structural configuration shown in Fig. 39 in the following way:

The geometrical structure of the central region $C_0$ is specified by the following keywords 'Atoms.Number' and 'Atoms.SpeciesAndCoordinates':

```
Atoms.Number        18
<Atoms.SpeciesAndCoordinates
  1  C  3.000  0.000  0.000  2.0 2.0
   .....
 18  C 28.500  0.000  0.000  2.0 2.0
Atoms.SpeciesAndCoordinates>
```

The geometrical structure of the left lead region $L_0$ is specified by the following keywords 'LeftLeadAtoms.Number' and 'LeftLeadAtoms.SpeciesAndCoordinates':

```
LeftLeadAtoms.Number         3
<LeftLeadAtoms.SpeciesAndCoordinates
  1  C -1.500  0.000  0.000  2.0 2.0
  2  C  0.000  0.000  0.000  2.0 2.0
  3  C  1.500  0.000  0.000  2.0 2.0
LeftLeadAtoms.SpeciesAndCoordinates>
```

The geometrical structure of the right lead region $R_0$ is specified by the following keywords 'RightLeadAtoms.Number' and 'RightLeadAtoms.SpeciesAndCoordinates'

```
RightLeadAtoms.Number        3
<RightLeadAtoms.SpeciesAndCoordinates
  1  C 30.000  0.000  0.000  2.0 2.0
  2  C 31.500  0.000  0.000  2.0 2.0
  3  C 33.000  0.000  0.000  2.0 2.0
RightLeadAtoms.SpeciesAndCoordinates>
```

This is the case of carbon chain which is demonstrated in the previous subsection. The central region $C_0$ is formed by 18 carbon atoms, and the left and right regions $L_0$ and $R_0$ contains three carbon atoms, respectively, where every bond length is 1.5 Å. Following the geometrical specification of device and leads, OpenMX will construct an extended central region $C(\equiv L_0|C_0|R_0)$ as shown in Fig. 39. The Green function for the extended central region $C$ is self-consistently determined in order to take account of relaxation of electronic structure around the interface between the central region $C_0$ and the region $L_0(R_0)$. In addition, we impose two conditions so that the central Green function can be calculated in the NEGF method [73]:

1. The localized basis orbitals $\phi$ in the region $C_0$ overlap with those in the regions $L_0$ and $R_0$, but do not overlap with those in the regions $L_1$ and $R_1$.

2. The localized basis orbitals $\phi$ in the $L_i$ ($R_i$) region has no overlap with basis orbitals in the cells beyond the nearest neighboring cells $L_{i-1}$ ($R_{i-1}$) and $L_{i+1}$ ($R_{i+1}$).

In our implementation the basis functions are strictly localized in real space because of the generation of basis orbitals by a confinement scheme [41, 42]. Therefore, once the localized basis orbitals with specific cutoff radii are chosen for each region, the two conditions can be always satisfied by just adjusting the size of the unit cells for $L_i$ and $R_i$.

Although the specification of unit cells for the regions $L_0$, $C_0$, and $R_0$ is not required, it should be noted that some periodicity is implicitly assumed. The construction of infinite leads is made by employing the unit cells used in the band structure calculations by the step 1, and the informations are stored in a file '*NEGF.filename.hks*'. Also, due to the structural configuration shown in Fig. 39, the unit vectors on the **bc**-plane for the left and right leads should be consistent. Thus, the unit vector on the **bc**-plane for the extended central region $C$ is implicitly assumed to be same as that of the leads. Within the structural limitation, you can set up the structural configuration.

The unit in the specification of the geometrical structure can be given by

```
Atoms.SpeciesAndCoordinates.Unit   Ang # Ang|AU
```

In the NEGF calculation, either 'Ang' or 'AU' for 'Atoms.SpeciesAndCoordinates.Unit' is supported, but 'FRAC' is not.

How OpenMX analyzes the geometrical structure can be confirmed by the standard output as shown below:

```
<TRAN_Calc_GridBound>

*****************************************************
The extended cell consists of Left0-Center-Right0.
The cells of left and right reads are connected as.
...|Left2|Left1|Left0-Center-Right0|Right1|Right2...

Each atom in the extended cell is assigned as follows:
where '12' and '2' mean that they are in 'Left0', and
'12' has overlap with atoms in the Left1,
and '13' and '3' mean that they are in 'Right0', and
'13' has overlap with atoms in the 'Right1', and also
'1' means atom in the 'Center'.
*****************************************************

Atom1  = 12 Atom2  =  2 Atom3  =  1 Atom4  =  1 Atom5  =  1 Atom6  =  1 Atom7  =  1
Atom8  =  1 Atom9  =  1 Atom10 =  1 Atom11 =  1 Atom12 =  1 Atom13 =  1 Atom14 =  1
Atom15 =  1 Atom16 =  1 Atom17 =  1 Atom18 =  1 Atom19 =  1 Atom20 =  1 Atom21 =  3
Atom22 = 13
```

The atoms in the extended cell consisting of $L_0|C_0|R_0$ are assigned by the numbers, where '12' and '2' mean that they are in $L_0$, and '12' has overlap with atoms in $L_1$, and '13' and '3' mean that they are in $R_0$, and '13' has overlap with atoms in $R_1$, and also '1' means atom in $C_0$. By checking the analysis you may confirm whether the structure is properly constructed or not.

### B. Keywords

The NEGF calculation of the step 2 is performed by the keyword 'scf.EigenvalueSolver':

```
scf.EigenvalueSolver       NEGF
```

For the NEGF calculation the following keywords are newly added.

```
NEGF.filename.hks.l     lead-chain.hks
NEGF.filename.hks.r     lead-chain.hks

NEGF.Num.Poles             100       # defalut=150
NEGF.scf.Kgrid             1 1       # defalut=1 1

NEGF.bias.voltage          0.0       # default=0.0 (eV)
NEGF.bias.neq.im.energy    0.01      # default=0.01 (eV)
NEGF.bias.neq.energy.step  0.02      # default=0.02 (eV)
```

An explanation for each keyword is given below.

```
NEGF.filename.hks.l     lead-chain.hks
NEGF.filename.hks.r     lead-chain.hks
```

The files containing information of leads are specified by the above two keywords, where 'NEGF.filename.hks.l' and 'NEGF.filename.hks.r' are for the left and right leads, respectively.

```
NEGF.Num.Poles             100       # defalut=150
```

The equilibrium density matrix is evaluated by a contour integration method [73, 74]. The number of poles used in the method is specified by the keyword 'NEGF.Num.Poles'.

```
NEGF.scf.Kgrid             1 1       # defalut=1 1
```

The numbers of **k**-points to discretize the reciprocal vectors $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{c}}$ are specified by the keyword 'NEGF.scf.Kgrid'. Since no periodicity is assumed along the **a**-axis, you do not need to specify that for the **a**-axis.

```
NEGF.scf.Iter.Band          6        # defalut=6
```

It would be better to use the conventional diagonalization method for a few SCF steps in the initial SCF iterations by assuming a periodicity along the **a**-axis as well as **b**- and **c**-axes. The procedure is effective to avoid an erratic charge distribution which is a serious problem in the self-consistent NEGF method. The number of first SCF steps for which the conventional diagonalization method is applied is controlled by the keyword 'NEGF.scf.Iter.Band'. Up to and including the SCF steps specified by 'NEGF.scf.Iter.Band', the conventional diagonalization method is used and then onward, the solver is switched from the conventional method to the NEGF method. The default is 6.

```
NEGF.bias.voltage          0.0       # default=0.0 (eV)
```

The source-drain bias voltage applied to the left and right leads is specified by the keyword 'NEGF.bias.voltage' in units of eV, corresponding to Volt. Noting that only the difference between applied bias voltages has physical meaning, you only have to give a single value as the source-drain bias voltage.

```
NEGF.bias.neq.im.energy    0.01      # default=0.01 (eV)
NEGF.bias.neq.energy.step  0.02      # default=0.02 (eV)
```

When a finite source-drain bias voltage is applied, a part of the density matrix is contributed by the non-equilibrium Green function. Since the non-equilibrium Green function is not analytic in general in the complex plane, the contour integration method used for the equilibrium Green function cannot be applied. Thus, in the current implementation the non-equilibrium Green function is evaluated on the real axis with a small imaginary part using a simple rectangular quadrature scheme. Then, the imaginary part is given by the keyword 'NEGF.bias.neq.im.energy' and the step width is given by the keyword 'NEGF.bias.neq.energy.step' in units of eV. In most cases, the default values are sufficient, while the detailed analysis of the convergence property can be found in Ref. [73]. How many energy points on the real axis are used for the evaluation of the non-equilibrium Green function can be confirmed in the standard output and the file '*System.Name*.out'. In case of 'NEGF-Chain.dat', if the bias voltage of 0.5 V is applied, you will see in the standard output that the energy points of 120 are allocated for the calculation as follows:

```
Intrinsic chemical potential (eV) of the leads
  Left lead:  -7.752843837400
  Right lead: -7.752843837400
  add voltage =  0.0000 (eV) to the  left lead: new ChemP (eV):  -7.7528
  add voltage =  0.5000 (eV) to the right lead: new ChemP (eV):  -7.2528

Parameters for the integration of the non-equilibrium part
  lower bound:           -8.706843837400 (eV)
  upper bound:           -6.298843837400 (eV)
  energy step:            0.020000000000 (eV)
  number of steps:           120
```

The total number of energy points where the Green function is evaluated is given by the sum of the number of poles and the number of energy points on the real axis determined by the two keywords 'NEGF.bias.neq.im.energy' and 'NEGF.bias.neq.energy.step', and you should notice that the computational time is proportional to the total number of energy points.

```
NEGF.Poisson.Solver       FD     # FD|FFT, default=FD
```

In the NEGF method, the electrostatic potential is calculated by either a finite difference plus two dimensional FFT (FD) [73] or three dimensional FFT (FFT) [75]. The choice of the Poisson solver is specified by the keyword 'NEGF.Poisson.Solver'. Both the methods provide similar electrostatic potentials for non-polar systems, while the difference can be large for polar systems. The former is a proper choice in a sense that the eletrostatic potential at the boundaries between the leads and the central region should be the same as that in the calculations of the step 1 for the leads, while the SCF convergence seems to be rather easily obtained by the latter. The default is FD.

### C. SCF criterion

In the NEGF method, the SCF criterion given by the keyword 'scf.criterion' is applied to the residual norm between the input and output charge densities 'NormRD', while in the other cases 'dUele' is monitored. See also the keyword 'NEGF.scf.Iter.Band'.

**D. Gate bias voltage**

In our implementation, the gate voltage $V_{\rm g}(x)$ is treated by adding an electric potential defined by

$$V_{\rm g}(x) = V_{\rm g}^{(0)} \exp\left[-\left(\frac{x-x_{\rm c}}{d}\right)^8\right],$$

where $V_{\rm g}^{(0)}$ is a constant value corresponding to the gate voltage, and is specified by the keyword 'NEGF.gate.voltage' as follows:

```
NEGF.gate.voltage   1.0    # default=0.0 (in eV)
```

$x_c$ the center of the region $C_0$, and $d$ the length of the unit vector along **a**-axis for the region $C_0$. Due to the form of the equation, the applied gate voltage affects mainly the region $C_0$ in the central region $C$. The electric potential may resemble the potential produced by the image charges [76].

**E. Density of States (DOS)**

In the NEGF calculation, the density of states can be calculated by setting the following keywords:

```
Dos.fileout                 on              # on|off, default=off
NEGF.Dos.energyrange     -15.0 25.0 5.0e-3  #default=-10.0 10.0 5.0e-3 (eV)
NEGF.Dos.energy.div        200              # default=200
NEGF.Dos.Kgrid             1 1              # default=1 1
```

When you want to calculate DOS, the keyword 'Dos.fileout' should be set to 'on' as usual. Also, the energy range where DOS is calculated is given by the keyword 'NEGF.Dos.energyrange', where the first and second numbers correspond to the lower and upper bounds, and the third number is an imaginary number used for smearing out DOS. The energy range specified by 'NEGF.Dos.energyrange' is divided by the number specified by the keyword 'NEGF.Dos.energy.div'. The numbers of **k**-points to discretize the reciprocal vectors $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{c}}$ are specified by the keyword 'NEGF.Dos.Kgrid'. The set of numbers given by 'NEGF.Dos.Kgrid' can be set larger than that by 'NEGF.scf.Kgrid' to increase computational accuracy. After the NEGF calculation with these parameters, you will find two files '*System.Name*.Dos.val' and '*System.Name*.Dos.vec', and can analyze those by the same procedure as usual. Also, it should be noted that the origin of energy is set to the chemical potential of the **left** lead.

## 43.4 Step 3: The transmission, current (density), and eigenchannel

After the calculations of the steps 2 and 3, you can proceed the calculations of transmission, current (density), and eigenchannel.

### 43.4.1 Transmission, total current, and conductance

At first, openmx calculates the transmission, the total current, and the conductance. The relevant keywords for the calculation are as follows:

```
NEGF.tran.Analysis          on         #  default on
NEGF.tran.CurrentDensity    on         #  default on
NEGF.tran.energyrange -10 10 1.0e-3  # default=-10.0 10.0 1.0e-3 (eV)
NEGF.tran.energydiv         200        # default=200
NEGF.tran.Kgrid             1 1        # default= 1 1
```

- NEGF.tran.Analysis, NEGF.tran.Channel, NEGF.tran.CurrentDensity

  If NEGF.tran.Analysis is set to on, the transmission, the total current, and the conductance are calculated.

- NEGF.tran.energyrange, NEGF.tran.energydiv

  The energy range where the transmission is calculated is given by the keyword 'NEGF.tran.energyrange', where the first and second numbers correspond to the lower and upper bounds, and the third number is an imaginary number used for smearing out the transmission. The energy range specified by 'NEGF.tran.energyrange' is divided by the number specified by the keyword 'NEGF.tran.energydiv'.

- NEGF.tran.Kgrid

  The numbers of **k**-points to discretize the reciprocal vectors $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{c}}$ are specified by the keyword 'NEGF.tran.Kgrid'. The set of numbers given by 'NEGF.tran.Kgrid' can be different and tends to be larger than that by 'NEGF.scf.Kgrid' to take account of the computational efficiency.

In the calculations of the transmission, the current, and the conductance, following messages are printed in the standard output.

```
*******************************************************
*******************************************************
 Welcome to TRAN_Main_Analysis.
 This is a post-processing code of OpenMX to analyze
 transport properties such as electronic transmission,
 current, eigen channel, and current distribution in
 real space based on NEGF.
 Copyright (C), 2002-2015, H. Kino and T. Ozaki
 TRAN_Main_Analysis comes with ABSOLUTELY NO WARRANTY.
 This is free software, and you are welcome to
 redistribute it under the constitution of the GNU-GPL.
*******************************************************
*******************************************************
```

```
Chemical potentials used in the SCF calculation
  Left lead:  -5.125617225230 (eV)
  Right lead: -5.125617225230 (eV)
NEGF.current.energy.step 1.0000e-02 seems to be large for the calculation of current ...
The recommended Tran.current.energy.step is 0.0000e+00 (eV).
  TRAN_Channel_kpoint  0    0.000000    0.000000
  TRAN_Channel_energy  0    0.000000 eV
  TRAN_Channel_Num 5

Parameters for the calculation of the current
  lower bound:     -5.125617225230 (eV)
  upper bound:     -5.125617225230 (eV)
  energy step:      0.010000000000 (eV)
  imaginary energy  0.001000000000 (eV)
  number of steps:   0

  calculating...

  myid0= 0 i2= 0 i3= 0  k2=  0.0000 k3= -0.0000
  myid0= 1 i2= 0 i3= 0  k2=  0.0000 k3= -0.0000

Transmission:  files

  ./negf-chain.tran0_0

Current:  file

  ./negf-chain.current


Conductance:  file

  ./negf-chain.conductance
```

After the calculations, in this case you will obtain three files `negf-chain.tran0_0`, `negf-chain.current`, `negf-chain.conductance`:

- *System.Name*.tran#_%

  The file stores transmissions for up- and down-spin states. The fourth column is the energy relative to the chemical potential of the **left** lead, and the sixth and eighth columns are transmission for up- and down-spin states, respectively. When you employ a lot of **k**-points which is given by 'NEGF.tran.Kgrid', a file with a different set of '#' and '%' in the file extension is generated for each **k**-point. The correspondence between the numbers and the **k**-points can be found in the file.

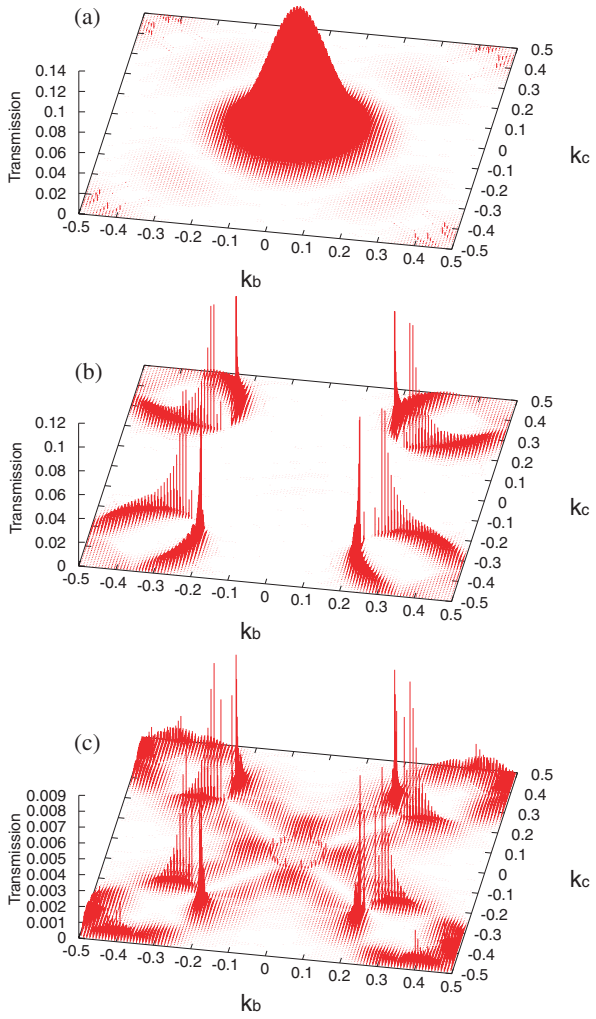



Figure 41: **k**-resolved transmission at the chemical potential for (a) the majority spin state of the parallel configuration, (b) the minority spin state of the parallel configuration, and (c) a spin state of the antiparallel configuration of Fe|MgO|Fe, respectively. For the calculations **k**-points of $120 \times 120$ were used.

- *System.Name*.current

  The file stores **k**-resolved currents and its average for up- and down-spin states in units of ampere.

- *System.Name*.conductance

  The file stores **k**-resolved conductance and its average for up- and down-spin states in units of quantum conductance ($G_0 \equiv \frac{e^2}{h}$). Thus, the conductance $G$ is proportional to the transmission $T$ at the chemical potential of the **left** lead, $\mu_L$, as follows:

$$G = \frac{e^2}{h} T(\mu_L)$$

As an example, the **k**-resolved transmission drawn by using the file '*System.Name*.conductance' is shown in Fig. 41.

### 43.4.2 Real-space charge/spin current density

In the step 3, real-space charge/spin current density is calculated optionally. The relevant keyword for the calculation is as follows:

```
NEGF.tran.CurrentDensity   on        #  default on
```

- `NEGF.tran.CurrentDensity`

  If `NEGF.tran.CurrentDensity` is set to `on`, the real-space current density is calculated.

In the calculation of the current density, `openmx` makes the following standard output:

```
Start Calculation of the currentdensity

  Spin #0
    Sum of current in real space [a.u.]
      Left(ideal) :    -9.10585e-06
      Right(ideal):    -9.10583e-06
      Left(truncated ):    -8.66971e-06
      Right(truncated):    -8.69926e-06
  Spin #1
    Sum of current in real space [a.u.]
      Left(ideal) :    -4.54540e-08
      Right(ideal):    -4.54544e-08
      Left(truncated ):    -4.19469e-08
      Right(truncated):    -4.27460e-08

Output: Currentdensity
  Charge-current density along a-axis: ./negf-8zgnr-0.3.curden1.cube
  Spin-current density along a-axis: ./negf-8zgnr-0.3.scurden1.cube
  Charge-current density: ./negf-8zgnr-0.3.curden.xsf
  Spin-current density: ./negf-8zgnr-0.3.scurden.xsf
  Voronoi Charge-current density: ./negf-8zgnr-0.3.curden_atom.xsf
  Voronoi Spin-current density: ./negf-8zgnr-0.3.scurden_atom.xsf
```

In this case, 6 files:
`negf-8zgnr-0.3.curden.xsf`, `negf-8zgnr-0.3.scurden.xsf`,
`negf-8zgnr-0.3.curden1.cube`, `negf-8zgnr-0.3.scurden1.cube`,
`negf-8zgnr-0.3.curden_atom.xsf`, `negf-8zgnr-0.3.scurden_atom.xsf`,
are generated. These files contain the following quantities:

- *System.Name*`.curden.xsf`, *System.Name*`.scurden.xsf`

  These files contain the charge and the spin current densities, respectively, on the real space grid. They can be visualized using 'Display→Forces' in XCrySDen.

- *System.Name*.curden1.cube, *System.Name*.scurden1.cube

  These files contain the **a**-component of the charge and the spin current densities, respectively, on real space grid. They can be visualized using VESTA, XCrySDen, and so on.

- *System.Name*.curden_atom.xsf, *System.Name*.scurden_atom.xsf

  These files contain the charge and the spin current density respectively averaged in Voronoi polyhedra of each atom. They can be visualized using 'Display→Forces' in XCrySDen.

(Experimentally) When you set

```
NEGF.OffDiagonalCurrent    on          #  default off
```

in an input file for the calculation with the non-collinear magnetism, the following files are outputted.

- *System.Name*.odcurden_r.xsf, *System.Name*.odcurden_i.xsf
- *System.Name*.odcurden1_r.cube, *System.Name*.odcurden1_i.cube
- *System.Name*.odcurden_atom_r.xsf, *System.Name*.odcurden_atom_i.xsf

These files contain the spin off-diagonal component of the current density. Since this quantity becomes generally a complex number, the real part and the imaginary part of that is output separately. As an example, we show in Fig. 42 the currentdensity in the 8-zigzag graphene nanoribbon with an antiferromagnetic junction under a finite bias voltage of 0.3 V. In the vicinity of boundaries, that shows an unphysical behavior because the basis set is not treated correctly in these regions. If you want to calculate more precisely the current density in these regions, please make a supercell larger.

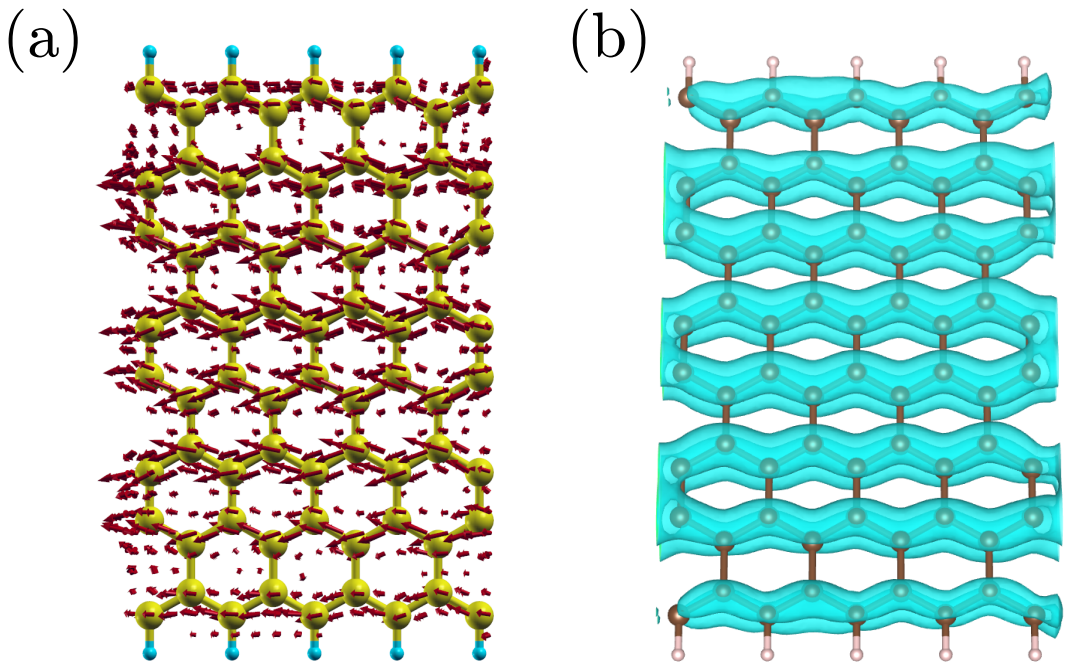



Figure 42: (a) The current density in the 8-zigzag graphene nanoribbon with an antiferromagnetic junction under a finite bias voltage of 0.3 V. (b) Isosurfaces of the **a**-compontent of that.

### 43.4.3 Eigenchannel analysis

In the step 3, transmission eigenchannels are calculated optionally. The relevant keywords for this calculation are as follows:

```
NEGF.tran.Channel          on        #  default on
NEGF.Channel.Nkpoint        1        # default=1
<NEGF.Channel.kpoint
  0.0  0.0
NEGF.Channel.kpoint>
# default 0.0 0.0
NEGF.Channel.Nenergy        1        # default=1
<NEGF.Channel.energy
  0.0
NEGF.Channel.energy>
# default 0.0
NEGF.Channel.Num    5    # defualt=5(for collinear), 10(for Non-collinear)
```

- `NEGF.tran.Channel`

  If `NEGF.tran.Channel` is set to `on`, the eigenchannel is calculated.

- `NEGF.Channel.Nkpoint`, `<NEGF.Channel.kpoint`, `NEGF.Channel.kpoint>`

  These keywords specify the **k** point, at which eigenchannels are calculated. Please write a **k** points per one line between `<NEGF.Channel.kpoint`and `NEGF.Channel.kpoint>`; the total number of **k** is `NEGF.Channel.Nkpoint`. The coordinate of the $k$ is two dimensional fractional coordinate; **k** should be specified as coefficients of two reciprocal lattice vector perpendicular to the transmittion direction.

- `NEGF.Channel.Nenergy`, `<NEGF.Channel.energy`, `NEGF.Channel.energy>`

  These keywords specify the energy, at which eigenchannels are calculated. Please write a energy per one line between `<NEGF.Channel.energy` and `NEGF.Channel.energy>`; the total number of energies is `NEGF.Channel.Nenergy`. The unit should be [eV] and the energy should be measured from the electrochemical potential of the left lead.

- `NEGF.Channel.Num`

  It specifies the number of eigenchannels that are printed in the real space representation. In each **k**, energy, spin, `NEGF.Channel.Num` eigenchannels in descending order about transmission eigenvalues are printed in the Gaussian cube format; the real and imaginary part is printed separately.

In the calculation of eigenchannels, `openmx` makes the following standard output:

```
***********************************************
 Calculation of transmission eigenchannels starts
***********************************************
```

```
  File index : negf-8zgnr-0.3.traneval#k_#E_#spin negf-8zgnr-0.3.tranevec#k_#E_#spin

  myid0 =   0, #k :    0, N_{ort} / N_{nonort} : 380 / 380
  PE    0 generates ./negf-8zgnr-0.3.traneval0_0_0 . Sum(eigenval) :   0.031643
  PE    0 generates ./negf-8zgnr-0.3.traneval0_0_1 . Sum(eigenval) :   0.000508

  Eigenchannel calculation finished
  They are written in plottable files.
  File index : negf-8zgnr-0.3.tranec#k_#E_#spin_#branch_r.cube(.bin)
               negf-8zgnr-0.3.tranec#k_#E_#spin_#branch_i.cube(.bin)

  ./negf-8zgnr-0.3.tranec0_0_0_0_r.cube     ./negf-8zgnr-0.3.tranec0_0_0_0_i.cube
  ./negf-8zgnr-0.3.tranec0_0_0_1_r.cube     ./negf-8zgnr-0.3.tranec0_0_0_1_i.cube
  ./negf-8zgnr-0.3.tranec0_0_0_2_r.cube     ./negf-8zgnr-0.3.tranec0_0_0_2_i.cube
  ./negf-8zgnr-0.3.tranec0_0_0_3_r.cube     ./negf-8zgnr-0.3.tranec0_0_0_3_i.cube
  ./negf-8zgnr-0.3.tranec0_0_0_4_r.cube     ./negf-8zgnr-0.3.tranec0_0_0_4_i.cube
  ./negf-8zgnr-0.3.tranec0_0_1_0_r.cube     ./negf-8zgnr-0.3.tranec0_0_1_0_i.cube
  ./negf-8zgnr-0.3.tranec0_0_1_1_r.cube     ./negf-8zgnr-0.3.tranec0_0_1_1_i.cube
  ./negf-8zgnr-0.3.tranec0_0_1_2_r.cube     ./negf-8zgnr-0.3.tranec0_0_1_2_i.cube
  ./negf-8zgnr-0.3.tranec0_0_1_3_r.cube     ./negf-8zgnr-0.3.tranec0_0_1_3_i.cube
  ./negf-8zgnr-0.3.tranec0_0_1_4_r.cube     ./negf-8zgnr-0.3.tranec0_0_1_4_i.cube
```

In this case, 22 files, `negf-8zgnr-0.3.treval0_0_0`, `negf-8zgnr-0.3.tranevec0_0_0`, `negf-8zgnr-0.3.tranec0_0_0_0_r.cube` - `negf-8zgnr-0.3.tranec0_0_1_4_r.cube`, `negf-8zgnr-0.3.tranec0_0_0_0_i.cube` - `negf-8zgnr-0.3.tranec0_0_1_4_i.cube`, are generated.

- *System.Name*`.traneval{#k}_{#E}_{#s}`

  This file contains transmission eigenvalues of all eigenchannels in the `{#k}`th **k**, `{#E}`th energy, and `{#s}`th spin.

- *System.Name*`.tranevec{#k}_{#E}_{#s}`

  This file contains LCAO components of all eigenchannels in the `{#k}`th **k**, `{#E}`th energy, and `{#s}`th spin.

  e. g. / `negf-chain.tranevec0_0_0`

```
************************************************************
************************************************************
        Eigenvalues and LCAO coefficients
        at the k-points specified in the input file.
************************************************************
************************************************************


   # of k-point = 0
   k2=   0.00000 k3=   0.00000
```

```
# of Energy = 0
e=   0.00000

Spin = Up

Real (Re) and imaginary (Im) parts of LCAO coefficients
```

| | | 1 | | 2 | | 3 | | 4 | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.9778 | | 0.0000 | | 0.0000 | | 0.0000 | |
| | | Re | Im | Re | Im | Re | Im | Re | Im |
| 1 | C 0 s | -0.00000 | -0.00000 | -0.00000 | 0.00000 | 0.00000 | 0.00000 | -0.00000 | 0.00000 |
| | 1 s | -0.00000 | -0.00000 | -0.00000 | -0.00000 | 0.00000 | -0.00000 | 0.00000 | -0.00000 |
| | 0 px | -0.63002 | -1.49377 | -0.14466 | 0.00019 | 0.01644 | -0.00032 | -0.07885 | 0.00095 |
| | 0 py | 0.00000 | 0.00000 | 0.00000 | 0.00000 | -0.00000 | 0.00000 | 0.00000 | 0.00000 |
| | 0 pz | 0.00000 | -0.00000 | 0.00000 | -0.00000 | -0.00000 | -0.00000 | -0.00000 | -0.00000 |
| 2 | C 0 s | -0.00000 | -0.00000 | -0.00000 | -0.00000 | 0.00000 | -0.00000 | 0.00000 | -0.00000 |
| | 1 s | 0.00000 | 0.00000 | -0.00000 | -0.00000 | -0.00000 | 0.00000 | -0.00000 | 0.00000 |
| | 0 px | 0.18040 | -0.03816 | -0.00452 | 0.00009 | -0.00545 | -0.00010 | -0.01970 | -0.00004 |
| | 0 py | -0.00000 | -0.00000 | -0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| | 0 pz | 0.00000 | -0.00000 | -0.00000 | 0.00000 | -0.00000 | -0.00000 | -0.00000 | -0.00000 |
| 3 | C 0 s | 0.00000 | 0.00000 | 0.00000 | -0.00000 | -0.00000 | 0.00000 | -0.00000 | 0.00000 |
| | 1 s | -0.00000 | -0.00000 | 0.00000 | 0.00000 | 0.00000 | -0.00000 | 0.00000 | -0.00000 |
| | 0 px | 2.06634 | 0.40490 | 0.11067 | 0.00023 | -0.06068 | 0.00009 | -0.06690 | -0.00042 |
| | 0 py | 0.00000 | 0.00000 | 0.00000 | -0.00000 | -0.00000 | 0.00000 | -0.00000 | 0.00000 |
| | 0 pz | 0.00000 | 0.00000 | 0.00000 | 0.00000 | -0.00000 | -0.00000 | 0.00000 | -0.00000 |
| 4 | C 0 s | 0.00000 | 0.00000 | 0.00000 | 0.00000 | -0.00000 | -0.00000 | -0.00000 | -0.00000 |
| | 1 s | -0.00000 | -0.00000 | -0.00000 | -0.00000 | -0.00000 | 0.00000 | -0.00000 | 0.00000 |

- *System.Name*`.tranec{#k}_{#E}_{#s}_{#c}_r.cube`,
  *System.Name*`.tranec{#k}_{#E}_{#s}_{#c}_i.cube`

  This file contains the real or the imaginary part of the eigenchannel in the Gaussian cube format. We can display isosurfaces from this files by using VESTA, XCrysDen, and so on. As an example, we show in Fig. 43 eigenchannels in the 8-zigzag graphene nanoribbon with an antiferromagnetic junction under a finite bias voltage of 0.3 V.

### 43.5 Running again the step 3 only

We can skip the NEGF calculation if it is finished in the previous `openmx` execution. In this case, please specify the following keyword:

```
NEGF.tran.SCF.skip  on
```

`openmx` reads a file *System.Name*`.tranb` generated in the previous step 2 calculation, and calculates the transmission, eigenchannels, and so on.

On the other hand, if we intend to stop `openmx` after it finishes the step 2 calculation, the following keywords should be set as follows:

```
NEGF.tran.SCF.skip  off
```

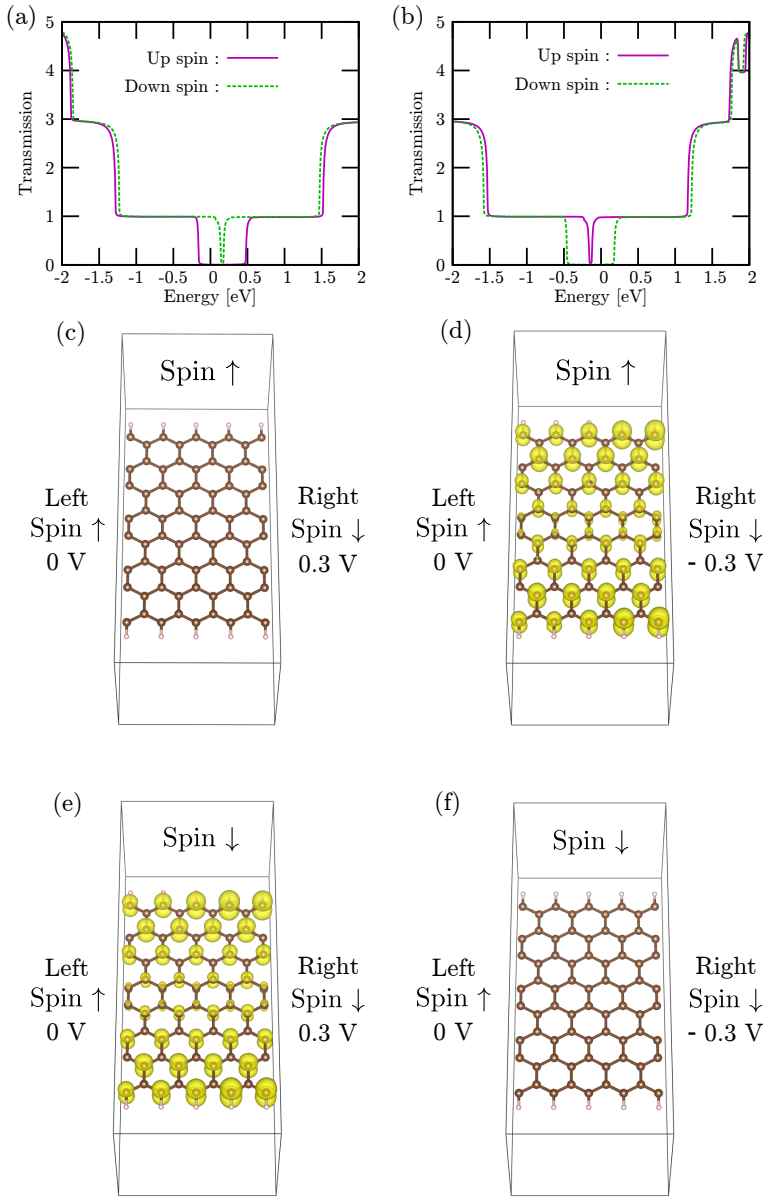



Figure 43: (a) Spin dependent transmission under a bias voltage of 0.3 V, (b) spin dependent transmission under a bias voltage of -0.3 V, (c) an eigenchannel at an energy of 0 eV, spin ↑, and 0.3 V as a bias voltage, (d) an eigenchannel at an energy of 0 eV, spin ↑, and - 0.3 V as a bias voltage, (e) an eigenchannel at an energy of 0 eV, spin ↓, and 0.3 V as a bias voltage, and (f) an eigenchannel at an energy of 0 eV, spin ↓, and - 0.3 V as a bias voltage in 8-zigzag graphene nanoribbon with an antiferromagnetic junction (The spin is ↑ in the left region and it is ↓ in the right region.) are depicted. The level of isosurfaces are identical in these figures; when the transmission is small, the eigenchannel itself is also small.

```
NEGF.tran.Analysis  off
NEGF.tran.Channel   off
```

### 43.6 Periodic system under zero bias

When the transmission of a system with the periodicity along the **a**-axis as well as the periodicity of the **bc**-plane is evaluated under zero bias voltage, it can be easily obtained by making use of the Hamiltonian calculated by the conventional band structure calculation without employing the Green function method. This scheme enables us to explore transport properties for a wide variety of possible geometric and magnetic structures with a low computational cost, and thereby can be very useful for many materials such as superlattice structures. The calculation is performed by adding a keyword 'NEGF.Output.for.TranMain':

```
NEGF.Output.for.TranMain    on
```

in the band structure calculation of the step 1. Then, after the calculation of the step 1, you will obtain a file '*System.Name*.tranb' which can be used in the calculation of the step 3, which means that you can skip the calculation of the step 2.

### 43.7 Interpolation of the effect by the bias voltage

Since for large-scale systems it is very time-consuming to perform the SCF calculation at each bias voltage, an interpolation scheme is available to reduce the computational cost in the calculations by the NEGF method. The interpolation scheme is performed in the following way: (i) the SCF calculations are performed for a few bias voltages which are selected in the regime of the bias voltage of interest. (ii) when the transmission and current are calculated, linear interpolation is made for the Hamiltonian block elements, $H^{(\mathbf{k})}_{\sigma,C}$ and $H^{(\mathbf{k})}_{\sigma,R}$, of the central scattering region and the right lead, and the chemical potential, $\mu_R$, of the right lead by

$$\begin{aligned} H^{(\mathbf{k})}_{\sigma,C} &= \lambda H^{(\mathbf{k},1)}_{\sigma,C} + (1-\lambda) H^{(\mathbf{k},2)}_{\sigma,C}, \\ H^{(\mathbf{k})}_{\sigma,R} &= \lambda H^{(\mathbf{k},1)}_{\sigma,R} + (1-\lambda) H^{(\mathbf{k},2)}_{\sigma,R}, \\ \mu_R &= \lambda \mu^{(1)}_R + (1-\lambda)\mu^{(2)}_R, \end{aligned}$$

where the indices 1 and 2 in the superscript mean that the quantities are calculated or used at the corresponding bias voltages where the SCF calculations are performed beforehand. In general, $\lambda$ should range from 0 to 1 for the moderate interpolation.

In the calculation of the step 3, the interpolation is made by adding the following keywords in the input file:

```
NEGF.tran.interpolate         on               # default=off, on|off
NEGF.tran.interpolate.file1  c1-negf-0.5.tranb
NEGF.tran.interpolate.file2  c1-negf-1.0.tranb
NEGF.tran.interpolate.coes    0.7 0.3          # default=1.0 0.0
```

When you perform the interpolation, the keyword 'NEGF.tran.interpolate' should be 'on'. In this case, files 'c1-negf-0.5.tranb' and 'c1-negf-1.0.tranb' specified by the keywords 'NEGF.tran.interpolate.file1' and 'NEGF.tran.interpolate.file2' are the results under bias voltages of 0.5 and 1.0 V, respectively, and

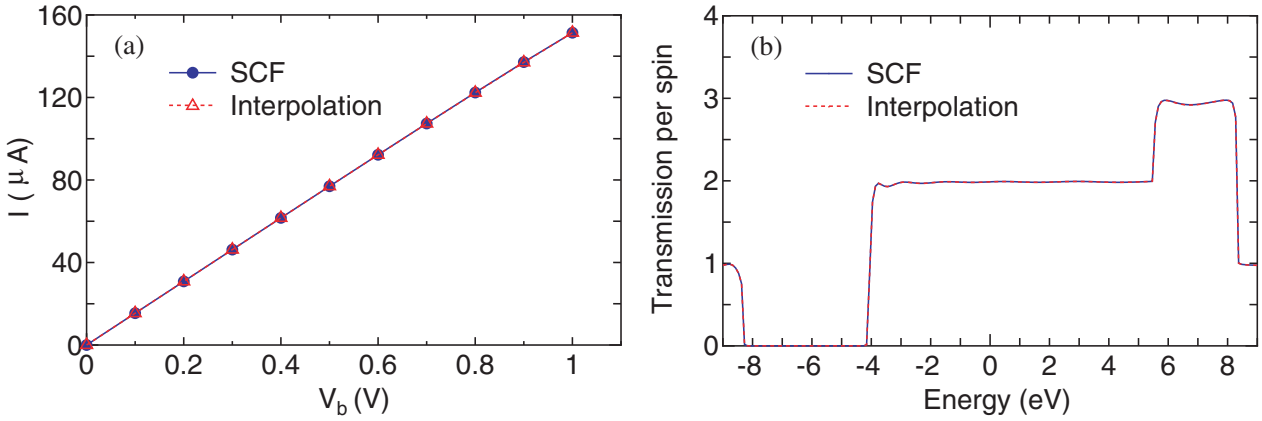



Figure 44: (a) Currents of the linear carbon chain calculated by the SCF calculations (solid line) and the interpolation scheme (dotted line). (b) Transmission of the linear carbon chain under a bias voltage of 0.3 V, calculated by the SCF calculations (solid line) and the interpolation scheme (dotted line). The imaginary part of 0.01 eV and the grid spacing of 0.01 eV are used for the integration of the nonequilibrium term in the density matrix.

the transmission and current at $V = 0.7 * 0.5 + 0.3 * 1.0 = 0.65[V]$ are evaluated by the interpolation scheme, where the weights of 0.7 and 0.3 are specified by the keyword 'NEGF.tran.interpolate.coes'.

A comparison between the fully self consistent and the interpolated results is shown with respect to the current and transmission in the linear carbon chain in Figs. 44(a) and (b). In this case, the SCF calculations at three bias voltages of 0, 0.5, and 1.0 V are performed, and the results at the other bias voltages are obtained by the interpolation scheme. For comparison we also calculate the currents via the SCF calculations at all the bias voltages. It is confirmed that the simple interpolation scheme gives notably accurate results for both the calculations of the current and transmission. Although the proper selection of bias voltages used for the SCF calculations may depend on systems, the result suggests that the simple scheme is very useful to interpolate the effect of the bias voltage while keeping the accuracy of the calculations.

### 43.8 Parallelization of NEGF

In the current implementation the NEGF calculation is parallelized by MPI. In addition to the MPI parallelization, if you use MKL, the matrix multiplication and the inverse calculation of matrix in the evaluation of the Green function are also parallelized by OpenMP. In this case, you can perform a hybrid parallelization by MPI/OpenMP which may lead to shorter computational time. The way for the parallelization is completely same as before.

In Fig. 45 we show the speed-up ratio in the elapsed time for the evaluation of the density matrix of 8-zigzag graphene nanoribbon (ZGNR) under a finite bias voltage of 0.5 eV. The energy points of 197 (101 and 96 for the equilibrium and nonequilibrium terms, respectively) are used for the evaluation of the density matrix. Only the $\Gamma$ point is employed for the **k**-point sampling, and the spin polarized calculation is performed. Thus, the combination of 394 for the three indices are parallelized by MPI. It is found that the speed-up ratio of the flat MPI parallelization, corresponding to 1 thread, reasonably scales up to 64 processes. Furthermore, it can be seen that the hybrid parallelization, corresponding to 2 and 4 threads, largely improves the speed-up ratio. By fully using 64 quad core processors,

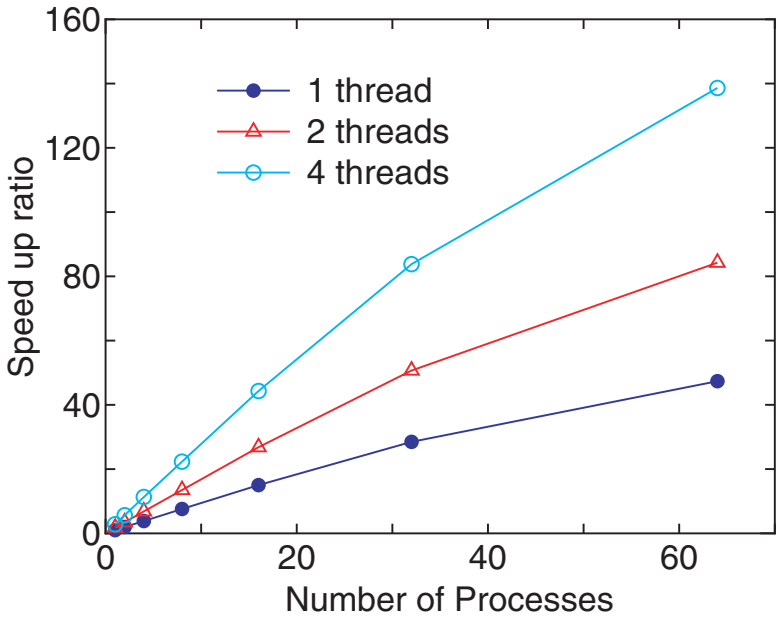



Figure 45: Speed-up ratio in the parallel computation of the calculation of the density matrix for the FM junction of 8-zigzag graphene nanoribbon (ZGNR) by a hybrid scheme using MPI and OpenMP. The speed-up ratio is defined by $T_1/T_p$, where $T_1$ and $T_p$ are the elapsed times by a single core and a parallel calculations. The cores used in the MPI and OpenMP parallelizations are called *process* and *thread*, respectively. The parallel calculations were performed on a CRAY-XT5 machine consisting of AMD opteron quad core processors (2.3GHz). In the benchmark calculations, the number of processes is taken to be equivalent to that of processors. Therefore, in the parallelization using 1 or 2 threads, 3 or 2 cores are idle in a quad core processor.

corresponding to 64 processes and 4 threads, the speed-up ratio is about 140, demonstrating the good scalability of the NEGF method. For the details see also Ref. [73]. It should be also noted that the number of processes in the MPI parallelization can exceed the number of atoms in OpenMX Ver. 3.9.

### 43.9 NEGF method for the non-collinear DFT

OpenMX Ver. 3.9 supports the NEGF method coupled with the non-collinear DFT method, which can be regarded as a full implementation of NEGF within NC-DFT. The spin-orbit coupling, the DFT+$U$ method, and the constraint schemes to control direction of spin and orbital magnetic moments supported for NC-DFT are all compatible with the implementation of the NEGF method. Thus, it is expected that a wide variety of problems can be treated, such as transport through magnetic domains with spiral magnetic structure. The usage of the functionality is the same as that for the collinear DFT case.

As an example, we show a result for zigzag graphene nanoribbon calculated by the NEGF method coupled with NC-DFT in Fig. 46. It is assumed that spin moments at the zigzag edges align upward and rightward in the left and right leads, respectively. Those calculations were performed by the conventional NC band structure method with the constraint scheme as the step 1. Then, any constraint was not applied in the calculation of the step 2. After getting the SCF convergence in the step 2, it is found that the spin direction gradually rotates in the central region as shown in Fig. 46(a).

The calculations can be traced by input files 'Lead-L-8ZGNR-NC.dat', 'Lead-R-8ZGNR-NC.dat', and 'NEGF-8ZGNR-NC.dat' stored in the directory 'work/negf_example'. Also, you will find another example for input files of a gold chain in the same directory.

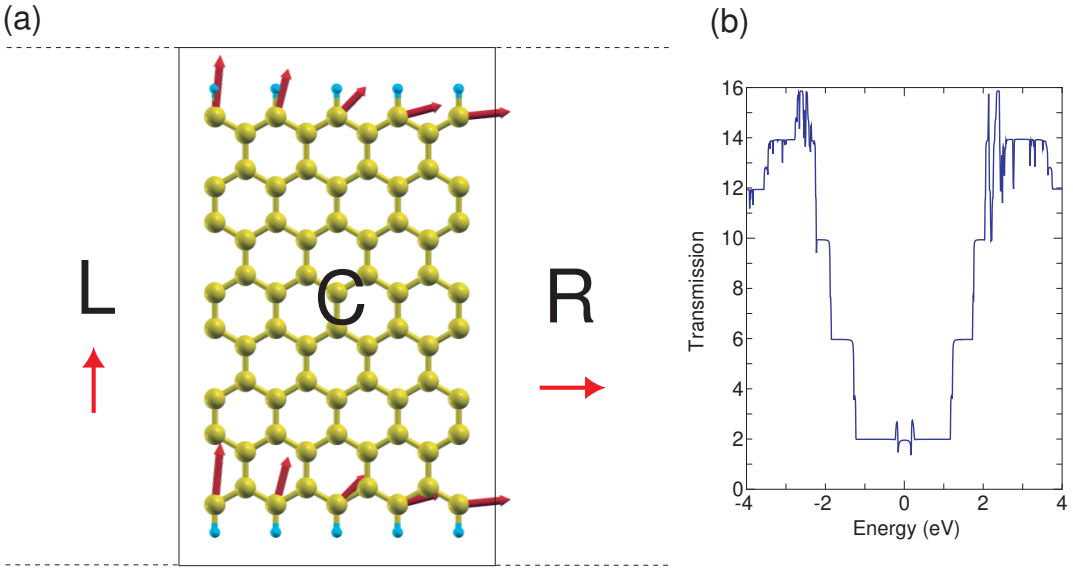



Figure 46: (a) Zigzag graphene nanoribbon with non-collinear spin direction represented by arrow. The length of the arrow corresponds to magnitude of the spin moment. In the calculations of the step 1, the constraint scheme to control spin direction was applied so that spin moments at the zigzag edges can align upward and rightward in the left and right leads, respectively. (b) Transmission of electron through the channel region C shown in Fig. 46(a).

### 43.10 Examples

For user's convenience, input files for five examples can be found in 'work/negf_example' as follows:

- Carbon chain under zero bias voltage

  Step 1: Lead-Chain.dat
  Step 2: NEGF-Chain.dat

- Graphene sheet under zero bias voltage

  Step 1: Lead-Graphene.dat
  Step 2: NEGF-Graphene.dat

- 8-zigzag graphene nanoribbon with an antiferromagnetic junction under a finite bias voltage of 0.3 V

  Step 1: Lead-L-8ZGNR.dat, Lead-R-8ZGNR.dat
  Step 2: NEGF-8ZGNR-0.3.dat

- 8-zigzag graphene nanoribbon with a non-collinear magnetic junction under zero bias

  Step 1: Lead-L-8ZGNR-NC.dat, Lead-R-8ZGNR-NC.dat
  Step 2: NEGF-8ZGNR-NC.dat

- Gold chain by NEGF coupled with NC-DFT under zero bias

  Step 1: Lead-Au-Chain-NC.dat
  Step 2: NEGF-Au-Chain-NC.dat

### 43.11 Automatic running test of NEGF

To check whether the NEGF calculation part is properly installed or not, an automatic running test for the NEGF calculation can be performed by

#### For the MPI parallel running

```
% mpirun -np 16 openmx -runtestNEGF
```

#### For the MPI/OpenMP parallel running

```
% mpirun -np 8 openmx -runtestNEGF -nt 2
```

Then, OpenMX will run with five test cases including calculations of the steps 1 and 2, and compare calculated results with the reference results which are stored in 'work/negf_example'. The comparison (absolute difference in the total energy, force, the averaged current density, the sum of the eigen transmission) is stored in a file 'runtestNEGF.result' in the directory 'work'. The reference results were calculated using 16 MPI processes of a 2.6GHz Xeon machine. If the difference is within last seven digits, we may consider that the installation is successful.

# 44 Maximally Localized Wannier Function

## 44.1 General

The following are descriptions on how to use OpenMX to generate maximally localized Wannier function (MLWF) [122, 123]. Keywords and settings for controlling the calculations are explained. The style of keywords are closely following those originally in OpenMX. Throughout the section, a couple of results for silicon in the diamond structure will be shown for convenience. The calculation can be traced by openmx code with an input file 'Si.dat' in 'work/wf_example'. There is no additional post processing code. After users may get the convergent result for the conventional SCF process for the electronic structure calculation, the following procedure explained below will be repeated by changing a couple of parameters with the restart file until desired MLWFs are obtained.

To acknowledge in any publications by using the functionality, the citation of the reference [77] would be appreciated:

### Switching on generating MLWFs

To switch on the calculation, a keyword 'Wannier.Func.Calc' should be explicitly set as 'on'. Its default value is 'off'.

```
Wannier.Func.Calc        on         #default off
```

### Setting the number of target MLWFs

The number of target MLWFs should be given explicitly by setting a keyword 'Wannier.Func.Num' and no default value for it.

```
Wannier.Func.Num       4         #no default
```

### Energy window for selecting Bloch states

The MLWFs will be generated from a set of Bloch states, which are selected by defining an energy window covering the eigenenergies of them. Following Ref. [123], two energy windows are introduced. One is so-called outer window, defined by two keywords, 'Wannier.Outer.Window.Bottom' and 'Wannier.Outer.Window.Top', indicating the lower and upper boundaries, respectively. The other one is inner window, which is specified by two similar key words, 'Wannier.Inner.Window.Bottom' and 'Wannier.Inner.Window.Top'. All these four values are given in the unit of eV relative to the Fermi level. The inner window should be fully inside of the outer window. If the two boundaries of inner window are equal to each other, it means inner window is not defined and not used in calculation. There is no default values for the outer window, while 0.0 is the default value for two boundaries of inner window. One example is as following:

```
Wannier.Outer.Window.Bottom   -14.0   #lower boundary of outer window, no default value
Wannier.Outer.Window.Top        0.0   #upper boundary of outer window, no default value
Wannier.Inner.Window.Bottom     0.0   #lower boundary of inner window, default value 0.0
Wannier.Inner.Window.Top        0.0   #upper boundary of outer window, default value 0.0
```

To set these two windows covering interested bands, it is usually to plot band structure and/or density of states before the calculation of MLWFs. If you want to restart the minimization of MLWFs by reading the overlap matrix elements from files, the outer window should not be larger than that used for calculating the stored overlap matrix. Either equal or smaller is allowed. The inner window can be varied within the outer window as you like when the restart calculation is performed. This would benefit the restarting calculation or checking the dependence of MLWFs on the size of both the windows. For the restarting calculation, please see also the section (7) 'Restart optimization without calculating overlap matrix'.

**Initial guess of MLWFs**

User can choose whether to use initial guess of target MLWFs or not by setting the keyword 'Wannier.Initial.Guess' as 'on' or 'off'. Default value is 'on', which means we recommend user to use an initial guess to improve the convergence or avoid local minima during the minimization of spread function.

If the initial guess is required, a set of local functions with the same number of target MLWFs should be defined. Bloch wave functions inside the outer window will be projected on to them. Therefore, these local functions are also called as projectors. The following steps are required to specify a projector.

*A. Define local functions for projectors*

Since the pseudo-atomic orbitals are used for projectors, the specification of them is the same as for the basis functions. An example setting, for silicon in diamond structure, is as following:

```
Species.Number          2

<Definition.of.Atomic.Species
  Si       Si7.0-s2p2d1    Si_CA19
  proj1    Si5.5-s1p1d1f1  Si_CA19
Definition.of.Atomic.Species>
```

In this example, since we employ PAOs from Si as projectors, an additional specie 'proj1' is defined as shown above. Inside the pair keywords '<Definition.of.Atomic.Species' and 'Definition.of.Atomic.Species>', in addition to the first line used for Si atoms, one species for the projectors is defined. Its name is 'proj1' defined by 'Si5.5-s1p1d1f1' and the pseudopotential 'Si_CA19'. In fact, the pseudopotential defined in this line will not be used. It is given just for keeping the consistency of inputting data structure. One can use any PAO as projector. Also the use of only a single basis set is allowed for each l-component. We strongly recommend user to specify 's1p1d1f1' in all cases to avoid possible errors.

*B. Specify the orbital, central position and orientation of a projector*

Pair keywords '<Wannier.Initial.Projectos' and 'Wannier.Initial.Projectos>' will be used to specify

the projector name, local orbital function, center of local orbital, and the local $z$-axis and $x$-axis for the orbital orientation.

An example setting is shown here:

```
<Wannier.Initial.Projectors
   proj1-sp3   0.250  0.250  0.250   -1.0 0.0 0.0    0.0  0.0 -1.0
   proj1-sp3   0.000  0.000  0.000    0.0 0.0 1.0    1.0  0.0  0.0
Wannier.Initial.Projectors>
```

Each line contains the following items. For example, in the first line, the species name, 'proj1', is defined in pairing keywords 'Definition.of.Atomic.Species'. '-' is used to connect the projector name and the selected orbitals. 'sp3' means that the sp3 hybridized orbitals of this species is used as the initial guess of four target Wannier functions (see also Table 6 for all the possible orbitals and their hybrids). The projectors consisting of hybridized orbitals are centered at the position given by the following 3 numbers, '0.25 0.25 0.25', which are given in unit defined by keyword 'Wannier.Initial.Projectors.Unit' (to be explained below). The next two sets of three numbers define the $z$-axis and $x$-axis of the local coordinate system, respectively, where each axis is specified by the vector defined by three components in $xyz$-coordinate. In this example, in the first line the local z-axis defined by '-1.0 0.0 0.0' points to the opposite direction to the original $x$-axis, while the local $x$-axis defined by '0.0 0.0 -1.0' points to the opposite direction to the original $z$-axis. In the second line the local axes are the same as the original coordinate system.

The orbital used as projector can be the original PAOs or any hybrid of them. One must be aware that the total number of projectors defined by 'sp3' is 4. Similarly, 'sp' and 'sp2' contain 2 and 3 projectors, respectively. A list of supported PAOs and hybridizations among them can be found in Table 9. Any name other than those listed is not allowed.

The projector can be centered anywhere inside the unit cell. To specify its location, we can use the fractional (FRAC) coordinates relative to the unit cell vectors or Cartesian coordinates in atomic unit (AU) or in angstrom (ANG). The corresponding keyword is 'Wannier.Initial.Projectors.Unit'.

```
   Wannier.Initial.Projectors.Unit     FRAC     #AU, ANG or FRAC
```

**K grid mesh and b vectors connecting neighboring k-points**

The Monkhorst-Pack **k** grid mesh is defined by the keyword 'Wannier.Kgrid'. There is no default setting for it. To use finite difference approach for calculating k-space differentials, **b** vectors connecting neighboring k points are searched shell by shell according to the distance from a central **k** point. The maximum number of searched shells is defined by keyword 'Wannier.MaxShells'. The default value is 12 and it should be increased if failure in finding a set of proper **b** vectors. The problem may happen in case of a system having a large aspect ratio among unit vectors, and in this case you will see an error message, while the value 12 works well in most cases. A proper setting of 'Wannier.Kgrid' will also help to find **b** vectors, where the grid spacing by the discretization for each reciprocal lattice vector should be nearly equivalent to each other.

```
   Wannier.MaxShells          12          # default value is 12.
   Wannier.Kgrid           8  8  8        # no default value
```

Table 9: Orbitals and hybrids used as projector. The hybridization is done within the new coordinate system defined by $z$- and $x$-axes.

| Orbital name | Number of included projector | Description |
|---|---|---|
| s | 1 | $s$ orbital from PAOs |
| p | 3 | $p_x, p_y, p_z$ from PAOs |
| px | 1 | $p_x$ from PAOs |
| py | 1 | $p_y$ from PAOs |
| pz | 1 | $p_z$ from PAOs |
| d | 5 | $d_{z^2}, d_{x^2-y^2}, d_{xy}, d_{xz}, d_{yz}$ from PAOs |
| dz2 | 1 | $d_{z^2}$ from PAOs |
| dx2-y2 | 1 | $d_{x^2-y^2}$ from PAOs |
| dxy | 1 | $d_{xy}$ from PAOs |
| dxz | 1 | $d_{xy}$ from PAOs |
| dyz | 1 | $d_{xy}$ from PAOs |
| f | 7 | $f_{z^3}, f_{xz^2}, f_{yz^2}, f_{zx^2}, f_{xyz}, f_{x^3-3xy^2}, f_{3yx^2-y^3}$ from PAOs |
| fz3 | 1 | $f_{z^3}$ from PAOs |
| fxz2 | 1 | $f_{xz^2}$ from PAOs |
| fyz2 | 1 | $f_{yz^2}$ from PAOs |
| fzx2 | 1 | $f_{zx^2}$ from PAOs |
| fxyz | 1 | $f_{xyz}$ from PAOs |
| fx3-3xy2 | 1 | $f_{x^3-3xy^2}$ from PAOs |
| f3yx2-y3 | 1 | $f_{3yx^2-y^3}$ from PAOs |
| sp | 2 | Hybridization between s and px orbitals, including $\frac{1}{\sqrt{2}}(s+p_x)$ and $\frac{1}{\sqrt{2}}(s-p_x)$ |
| sp2 | 3 | Hybridization among s, px, and py orbitals, including $\frac{1}{\sqrt{3}}s-\frac{1}{\sqrt{6}}p_x+\frac{1}{\sqrt{2}}p_y$, $\frac{1}{\sqrt{3}}s-\frac{1}{\sqrt{6}}p_x-\frac{1}{\sqrt{2}}p_y$ and $\frac{1}{\sqrt{3}}s+\frac{2}{\sqrt{6}}p_x$ |
| sp3 | 4 | Hybridization among s, px, py and pz orbitals: $\frac{1}{\sqrt{2}}(s+p_x+p_y+p_z)$, $\frac{1}{\sqrt{2}}(s+p_x-p_y-p_z)$ $\frac{1}{\sqrt{2}}(s-p_x+p_y-p_z)$, $\frac{1}{\sqrt{2}}(s-p_x-p_y+p_z)$ |
| sp3dz2 | 5 | Hybridization among $s, p_x, p_y, p_z$ and $d_{z^2}$ orbitals: $\frac{1}{\sqrt{3}}s-\frac{1}{\sqrt{6}}p_x+\frac{1}{\sqrt{2}}p_y$, $\frac{1}{\sqrt{3}}s-\frac{1}{\sqrt{6}}p_x+\frac{1}{\sqrt{2}}p_y$, $\frac{1}{\sqrt{3}}s-\frac{2}{\sqrt{6}}p_x$ $\frac{1}{\sqrt{2}}p_z+\frac{1}{\sqrt{2}}d_{z^2}, -\frac{1}{\sqrt{2}}p_z+\frac{1}{\sqrt{2}}d_{z^2}$ |
| sp3deg | 6 | Hybridization among $s, p_x, p_y, p_z$ and $d_{z^2}, d_{x^2-y^2}$ orbitals: $\frac{1}{\sqrt{6}}s-\frac{1}{\sqrt{2}}p_x-\frac{1}{\sqrt{12}}d_{z^2}+\frac{1}{2}d_{x^2-y^2}$, $\frac{1}{\sqrt{6}}s+\frac{1}{\sqrt{2}}p_x-\frac{1}{\sqrt{12}}d_{z^2}+\frac{1}{2}d_{x^2-y^2}$, $\frac{1}{\sqrt{6}}s-\frac{1}{\sqrt{2}}p_y-\frac{1}{\sqrt{12}}d_{z^2}-\frac{1}{2}d_{x^2-y^2}$, $\frac{1}{\sqrt{6}}s+\frac{1}{\sqrt{2}}p_y-\frac{1}{\sqrt{12}}d_{z^2}-\frac{1}{2}d_{x^2-y^2}$, $\frac{1}{\sqrt{6}}s-\frac{1}{\sqrt{2}}p_z+\frac{1}{\sqrt{3}}d_{z^2}, \frac{1}{\sqrt{6}}s+\frac{1}{\sqrt{2}}p_z+\frac{1}{\sqrt{3}}d_{z^2}$ |

**Minimizing spread of WF**

For entangled band case [123], two steps are needed to find the MLWFs. The first step is to minimize the gauge invariant part of spread function by disentangling the non-isolated bands. The second step is the same as isolated band case [122]. The gauge dependent part is optimized by unitary transformation of the selected Bloch wave functions according to the gradient of spread function. For the first step, three parameters are used to control the self-consistence loop. They are 'Wannier.Dis.SCF.Max.Steps', 'Wannier.Dis.Conv.Criterion', and 'Wannier.Dis.Mixing.Para'. They are the maximum number of SCF loops, the convergence criterion, and the parameter to control the mixing of input and output subspace projectors, respectively.

```
Wannier.Dis.SCF.Max.Steps         2000         # default 200
Wannier.Dis.Conv.Criterion        1e-12        # default 1e-8
Wannier.Dis.Mixing.Para           0.5          # default value is 0.5
```

For the second step, three minimization methods are available. One is a steepest decent (SD) method, and the second one is a conjugate gradient (CG) method. The third one is a hybrid method which uses the SD method firstly and then switches to the CG method. The keyword 'Wannier.Minimizing.Scheme' indicates which method to be used. '0', '1', and '2' mean the simple SD method, the CG method, and hybrid method, respectively. The step length for the SD method is set by the keyword 'Wannier.Minimizing.StepLength'. In the CG method, a secant method is used to determine the optimized step length. The maximum secant steps and initial step length is specified by 'Wannier.Minimizing.Secant.Steps' and 'Wannier.Minimizing.Secant.StepLength', respectively. The maximum number of minimization step and convergence criterion are controlled by 'Wannier.Minimizing.Max.Steps' and 'Wannier.Minimizing.Conv.Criterion', respectively.

```
Wannier.Minimizing.Scheme             2     # default 0, 0=SD 1=CG 2=hybrid
Wannier.Minimizing.StepLength        2.0    # default 2.0
Wannier.Minimizing.Secant.Steps       5     # default 5
Wannier.Minimizing.Secant.StepLength 2.0    # default 2.0
Wannier.Minimizing.Conv.Criterion   1e-12   # default 1e-8
Wannier.Minimizing.Max.Steps         200    # default 200
```

In the hybrid minimization scheme, SD and CG have the same number of maximum minimization steps as specified by 'Wannier.Minimizing.Max.Steps'.

**Restarting optimization without calculating overlap matrix**

If the overlap matrix $M_{mn}^{(\mathbf{k},\mathbf{b})}$ has been calculated and stored in a disk file, the keyword 'Wannier.Readin.Overlap.Matrix' can be set as 'on' to restart generating MLWF without calculating $M_{mn}^{(\mathbf{k},\mathbf{b})}$ again.

```
Wannier.Readin.Overlap.Matrix       off      # default is on
```

This can save the computational time since the calculation of overlap matrix is time consuming. The code will read the overlap matrix as well as the eigenenergies and states from the disk file. One should keep in mind that the outer window and k grid should be the same as those used for calculating

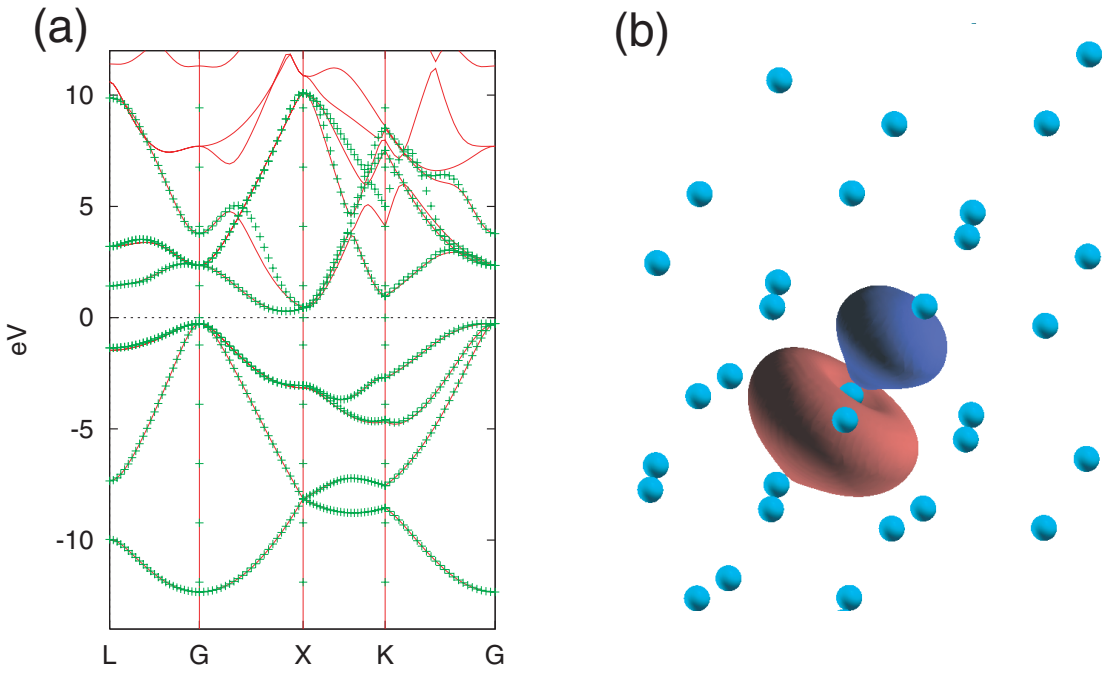



Figure 47: (a) The interpolated band structure (symbolic line) of Si in diamond structure is compared with the original band structure (solid line). (b) One of the eight converged MLWFs from four valence states and four conduction states near Fermi level of Si in diamond structure. It is obtained with an initial guess of sp3 hybrid.

the stored overlap matrix and eigenvalues. The consistency will be checked in the code. The inner window, initial guess of MLWF as well as the convergence criteria can be adjusted for restarting optimization. If 'Wannier.Readin.Overlap.Matrix' is set as 'off', the overlap matrix will be calculated and automatically stored into a disk file. The file name is defined by 'System.Name' with an file extension '.mmn'. The eigenenergies and states are also stored in the disk file with extension '.eigen'.

## 44.2 Analysis

### Plotting interpolated band structure

To plot the interpolated band structure, set 'Wannier.Interpolated.Bands' to be 'on'.

```
Wannier.Interpolated.Bands              on     # on|off, default=off
```

Other necessary settings, like k-path and sampling density along each path, are borrowed from those for plotting band dispersion in OpenMX. Therefore, the keyword 'Band.dispersion' should be set as 'on' in order to draw interpolated band structure. After convergence, interpolated band dispersion data will be found in a file with the extension name '.Wannier_Band', which has the same format as '.Band' file. As an example, the interpolated band structure of Si in diamond structure is shown together with its original band structure in Fig. 47(a).

**Plotting MLWF**

To plot the converged MLWFs, please change the keyword 'Wannier.Function.Plot' to be 'on'. The default value of it is 'off'.

```
Wannier.Function.Plot                   on          # default off
Wannier.Function.Plot.SuperCells      1 1 1         # default=0 0 0
```

If it is turned on, all the MLWFs will be plotted. They are written in the Gaussian cube file format with the file extension such as '.mlwf1_4_r.cube'. The file is named in the same style as HOMO or LUMO molecular orbitals files. The first number after '.mlwf' indicates the spin index and the following one are index of MLWFs and the last letter 'r' or 'i' means the real or imaginary part of the MLWF. Users can set the supercell size for plotting MLWF. It is defined by the keyword 'Wannier.Function.Plot.SuperCells'. '1 1 1' in the above example means that the unit cell is extended by one in both the plus and minus directions along the a-, b-, and c-axes by putting the home unit cell at the center, and therefore the MLWFs are plotted in an extended cell consisting of 27 (= (1∗2+1)∗(1∗2+1)∗(1∗2+1)) cells in this case. Figure 47(b) shows one of the eight converged MLWFs from four valence states and four conduction states near Fermi level of Si in diamond structure.

## 44.3 Monitoring optimization of spread function

The output during optimization steps is printed to the standard output. To monitor the optimization progress, the following method may be helpful. For convenient, we assume the standard output is stored in a file 'stdout.std'. The following example is for Si.dat which can be found in openmx*.*/work/wf_example, and each user can trace the same calculation.

**DISE**

Monitor the self-consistent loops for disentangling progress (the first step of optimization):

```
% grep "DISE" stdout.std

|  Iter  | Omega_I (Angs^2) | Delta_I (Angs^2) |  ---> DISE
|     1  |   18.371525257652|   18.371525257652|  ---> DISE
|     2  |   17.955767336391|   -0.415757921261|  ---> DISE
|     3  |   17.659503060694|   -0.296264275698|  ---> DISE
|     4  |   17.454033576174|   -0.205469484520|  ---> DISE
|     5  |   17.311180447271|   -0.142853128902|  ---> DISE
|     6  |   17.210945408916|   -0.100235038355|  ---> DISE
|     7  |   17.139778800398|   -0.071166608519|  ---> DISE
|     8  |   17.088603102826|   -0.051175697572|  ---> DISE
|     9  |   17.051329329614|   -0.037273773211|  ---> DISE
|    10  |   17.023842837298|   -0.027486492316|  ---> DISE
.........
.....
```

```
...
.
```

where 'Iter', 'Omega_I', and 'Delta_I' mean the iteration number, the gauge invariant part of the spread function, and its difference between two neighboring steps. The criterion given by the keyword 'Wannier.Dis.Conv.Criterion' is applied to 'Delta_I'.

**CONV**

Monitor the optimization of the gauge dependent part of the spread function (the second step of optimization):

```
% grep "CONV" stdout.std

Opt Step |Mode of Gradient|d_Omega_in_steps|     d_Omega    | (in Angs^2) ---> CONV
| SD    1 | 6.52434844E-01 | 5.41612774E-04 |-5.41340331E-04|  ---> CONV
| SD    2 | 6.51123660E-01 | 5.40524307E-04 |-5.40253165E-04|  ---> CONV
.....
.....
| SD  200 | 4.77499752E-01 | 3.96392019E-04 |-3.96271308E-04|  ---> CONV
|Opt Step |Mode of Gradient|     d_Omega    | (Angs^2) ---> CONV
| CG    1 | 8.61043764E-01 | -3.24716990E-01|  ---> CONV
.....
.....
| CG   58 | 1.67083857E-12 | -5.37225101E-13|  ---> CONV
| CG   59 | 5.44431651E-13 | -1.98972260E-13|  ---> CONV
************************************************************ ---> CONV
CONVERGENCE ACHIEVED !                        ---> CONV
************************************************************ ---> CONV
CONVERGENCE ACHIEVED !                        ---> SPRD
```

where 'Opt Step' and 'Modu.of Gradient' are the optimization step in either 'SD' or 'CG' method and the modulus of gradient of the spread function. The difference between two neighboring steps in the gauge dependent spread functions is calculated in two different way in the SD method, giving 'd_Omega_in_steps' and 'd_Omega'. 'd_Omega_in_steps' is given by

$$d\Omega = \epsilon \sum_{\mathbf{k}} ||G^{(\mathbf{k})}||^2,$$

where $\epsilon$ is the step length, $G^{(\mathbf{k})}$ is the gradient of the spread function. The details of the equation can be found in Ref. [122]. On the other hand, 'd_Omega' is given by

$$d\Omega = \Omega^{(n+1)} - \Omega^{(n)},$$

where $n$ is the iteration number. In the CG method, only 'd_Omega' is evaluated. The criterion given by the keyword 'Wannier.Minimizing.Conv.Criterion' is applied to 'Modu.of Gradient'.

### SPRD

Monitor the variation of spread of the Wannier functions:

```
% grep "SPRD" stdout.std

|Opt Step |     Omega_I     |     Omega_D     |     Omega_OD  |    Tot_Omega  | (in Angs^2) ---> SPRD
| SD    1 |   16.93053479  |    0.13727387  |    6.57748455 |   23.64529321 |  ---> SPRD
| SD    2 |   16.93053479  |    0.13724827  |    6.57696989 |   23.64475295 |  ---> SPRD
| SD    3 |   16.93053479  |    0.13722279  |    6.57645620 |   23.64421378 |  ---> SPRD
| SD    4 |   16.93053479  |    0.13719743  |    6.57594347 |   23.64367569 |  ---> SPRD
.....
.....
| SD  199 |   16.93053479  |    0.13399285  |    6.48989479 |   23.55442243 |  ---> SPRD
| SD  200 |   16.93053479  |    0.13398326  |    6.48950811 |   23.55402616 |  ---> SPRD
|Opt Step |     Omega_I     |     Omega_D     |     Omega_OD  |    Tot_Omega  | (Angs^2) ---> SPRD
| CG    1 |    16.93053479 |     0.15480701 |    6.14396737 |   23.22930917 |  ---> SPRD
| CG    2 |    16.93053479 |     0.17172507 |    5.87830203 |   22.98056189 |  ---> SPRD
| CG    3 |    16.93053479 |     0.17012089 |    5.78940789 |   22.89006357 |  ---> SPRD
.....
.....
| CG   57 |    16.93053479 |     0.16557875 |    5.73752928 |   22.83364282 |  ---> SPRD
| CG   58 |    16.93053479 |     0.16557876 |    5.73752928 |   22.83364282 |  ---> SPRD
| CG   59 |    16.93053479 |     0.16557876 |    5.73752928 |   22.83364282 |  ---> SPRD
************************************************************ ---> SPRD
CONVERGENCE ACHIEVED !                       ---> SPRD
************************************************************ ---> SPRD
```

where 'Opt Step' is the optimization step in either 'SD' or 'CG' method. 'Omega_I' is the gauge invariant part of spread function. 'Omega_D' and 'Omega_OD' are the gauge dependent diagonal and off-diagonal contribution, respectively. 'Tot_Omega' is the sum up of all the above three components of the spread function.

### CENT

Monitor the variation of Wannier function center:

```
% grep "CENT" stdout.std
WF   1 ( 1.14164289, 1.14164298, 1.14164266) |   2.95573380  --->CENT
WF   2 ( 1.55716251, 1.55716342, 1.14164203) |   2.95572597  --->CENT
WF   3 ( 1.55716191, 1.14164295, 1.55716190) |   2.95572978  --->CENT
WF   4 ( 1.14164389, 1.55716087, 1.55716055) |   2.95572957  --->CENT
WF   5 ( 0.20775982, 0.20775967, 0.20775893) |   2.95572677  --->CENT
WF   6 ( 0.20776045,-0.20775959,-0.20775914) |   2.95572605  --->CENT
WF   7 (-0.20775851, 0.20775981,-0.20775888) |   2.95572925  --->CENT
WF   8 (-0.20775787,-0.20775767, 0.20775933) |   2.95573335  --->CENT
Total Center ( 5.39761509, 5.39761243, 5.39760738) sum_spread  23.64583455 --->CENT
SD     1 ------------------------------------------------------------------------> CENT
WF   1 ( 1.14164582, 1.14164592, 1.14164559) |   2.95566613  --->CENT
WF   2 ( 1.55715957, 1.55716049, 1.14164497) |   2.95565831  --->CENT
WF   3 ( 1.55715897, 1.14164588, 1.55715897) |   2.95566211  --->CENT
WF   4 ( 1.14164683, 1.55715794, 1.55715761) |   2.95566190  --->CENT
```

```
WF   5 ( 0.20775689, 0.20775673, 0.20775599) |   2.95565910  --->CENT
WF   6 ( 0.20775752,-0.20775666,-0.20775620) |   2.95565838  --->CENT
WF   7 (-0.20775558, 0.20775687,-0.20775594) |   2.95566158  --->CENT
WF   8 (-0.20775493,-0.20775474, 0.20775639) |   2.95566569  --->CENT
Total Center ( 5.39761509, 5.39761243, 5.39760738) sum_spread  23.64529321 --->CENT
SD     2 ------------------------------------------------------------------------> CENT
.....
.....
CG    59 ------------------------------------------------------------------------> CENT
WF   1 ( 1.14585349, 1.14584696, 1.14584386) |   2.85421846  --->CENT
WF   2 ( 1.55295615, 1.55294970, 1.14584792) |   2.85422167  --->CENT
WF   3 ( 1.55296133, 1.14584610, 1.55295139) |   2.85421070  --->CENT
WF   4 ( 1.14584053, 1.55296761, 1.55296391) |   2.85417080  --->CENT
WF   5 ( 0.20356211, 0.20355857, 0.20355600) |   2.85418933  --->CENT
WF   6 ( 0.20355119,-0.20355008,-0.20355192) |   2.85422458  --->CENT
WF   7 (-0.20355306, 0.20355395,-0.20355905) |   2.85420611  --->CENT
WF   8 (-0.20355603,-0.20356000, 0.20355520) |   2.85420117  --->CENT
Total Center ( 5.39761571, 5.39761281, 5.39760730) sum_spread  22.83364282 --->CENT
```

where the optimization method and step are indicated by starting with 'SD' or 'CG'. Lines starting with 'WF' show the center of each Wannier function with ($x$, $y$, $z$) coordinates in Å unit. and its spread in Å$^2$. The sum up of all the Wannier functions center and spread are given in the line starting with 'Total Center'.

### 44.4 Examples for generating MLWFs

Examples for different materials are prepared in the installation directory: work/wf_example.

- Benzene.dat

  for generating six $p_z$-orbital like Wannier functions from benzene's six $\pi$ molecular orbitals.

- GaAs.dat

  for generating maximally localized Wannier functions from four valence bands of GaAs.

- Si.dat

  for generating eight Wannier functions by including both valence and conduction bands of Si. The initial guess is $sp_3$ hybrids.

- symGra.dat

  for generating the Wannier function for graphene sheet. The initial guess is $sp_2$ hybrids and $p_z$ orbitals on carbon atoms.

- pmCVO.dat

  for generating $t_{2g}$-like Wannier functions for cubic perovskite $CaVO_3$ without spin polarization calculation.

- NC_CVO.dat

  similar to the case of pmCVO.dat except for the inclusion of spin-orbit coupling.

- GaAs_NC.dat

  similar to the case of GaAs.dat but spin-orbit coupling is included.

- VBz.dat

  for generating Wannier functions for Vanadium-Benzene infinite chain, which is studied in Ref. [77].

### 44.5 Output files

Additional four files generated by the calculation are explained below. They have different extension names. '.mmn' file is for storing the overlap matrix elements $M_{mn}^{(\mathbf{k},\mathbf{b})}$. '.amn' is for the initial guess projection matrix element $A_{mn}^{(\mathbf{k})}$. '.eigen' is for the eigenenergies and eigenstates at each k point. The '.HWR' file is for the hopping integrals among MLWFs on a set of lattice vectors which lies in the Wigner-Seitz supercells conjugated with the sampled k grids. For restarting the optimization calculation, '.mmn' file will be read instead of written. More detailed information of the four files will be given below.

*A. File format of '.mmn' file*

This file structure is closely following that in Wannier90 [145]. The first line of this file is the description of the numbers in the second line. The numbers from left to right in the second line are the number ($N_{win}$) of included bands within the outer window, the number of k points, the number of **b** vectors, the number of spin component, respectively. The next lines are data blocks of $M_{mn}^{(\mathbf{k},\mathbf{b})}$. The most outer loop is for spin component. The next is the loop of **k** points and then **b** vectors. The most inner loops are the band index $n$ and $m$, respectively. In each block, the first line are 5 numbers. The first two numbers are the index of present **k** point and the index of neighboring point **k+b**, respectively. The next three numbers indicates in which unit cell **k+b** point lies. From the second line are the real and imaginary part of each matrix element. In each block, there are $N_{win} \times N_{win}$ complex numbers. An example file, generated by the input file 'Si.dat', is shown here:

```
Mmn_zero(k,b). band_num, kpt_num, bvector num, spinsize
          10          512            8            1
    1  512    0    0    0
    0.571090282808   -0.819911068319
    0.000031357498   -0.000045367307
   -0.000149292597    0.000215591228
   -0.003821911756    0.005522040495
    0.028616452988    0.019804944108
    0.003677357735    0.002544970842
   -0.006610037555   -0.004574771451
   -0.000950861169   -0.000658076633
   -0.000000008855    0.000000005272
    ........
    .....
    ...
```

.

*B. File format of '.amn' file*

This file structure is closely following that in Wannier90 [145]. The first line of the file is the description of the whole file. Obviously, the four numbers in the second line are the number ($N_{win}$) of bands within the outer window, the number of k points, the number of target MLWFs and the number of spin component, respectively. Similarly, the data blocks are written in loops. The most outer loop is spin component and then **k** points, target MLWFs and number of bands. As described in the first line of this file. In each block, the first three integers are the band index, the index of MLWFs and index of k points, respectively. The next are real and imaginary of that matrix element. An example file, generated by the input file 'Si.dat', is shown here:

```
Amn. Fist line BANDNUM, KPTNUM, WANNUM, spinsize. Next is m n k...
          10          512           8           1
    1    1    1    0.053943539299    0.000161703961
    2    1    1   -0.000525446164   -0.000000008885
    3    1    1    0.002498021589    0.000000084311
    ... ...
    ... ...
   10    1    1   -0.000000023582   -0.000000000069
    1    2    1    0.053943534952    0.000161703965
    2    2    1    0.033382665372    0.000000493665
    3    2    1   -0.051189536188   -0.000001480360
    ........
    .....
    ...
    .
```

*C. File format of '.eigen' file*

This file contains the eigenenergies and eigenstates at each k point. The first line is the Fermi level of system. The number of bands is indicated in the second line of the file. The next data are mainly in two parts. The first part is the eigenenergies and the second one is the corresponding eigenstates. In each part, the loop of spin component is the most outer one. The next loop is k points, followed by band index. For eigenstates, there is one more inner loop for the basis set. An example file, generated by the input file 'Si.dat', is shown here:

```
Fermi level -0.112747
Number of bands 10
    1    1   -0.566228100179
    2    1   -0.122518136808
    3    1   -0.122518129040
    4    1   -0.122518115949
    5    1   -0.026598417854
    ... ...
    ... ...
WF kpt 1 (0.00000000,0.00000000,0.00000000)
1 1   0.4790338281  -0.0014359768
1 2   0.0440709749  -0.0001321095
```

```
1 3  -0.0000003333  -0.0000000000
    ........
    .....
    ...
    .
```

*D. File format of '.HWR' file*

This file contains the hopping integrals between the $m$th MLWF, $|m, \mathbf{0}\rangle$, in the home unit cell and the $n$th MLWF, $|n, \mathbf{R}\rangle$, in the unit cell at $\mathbf{R}$. The matrix element $\langle m, \mathbf{0}|\hat{H}|n, \mathbf{R}\rangle$ is written in the following way. In '.HWR' file, the first line is just a description. The number of MLWFs, number of lattice vectors inside of Wigner-Seitz supercell are in the second and third lines, respectively. The unit cell vectors are given in the fifth, sixth, and seventh lines. Spin polarization, whether it is a non-spin polarized calculation or a spin polarized one with collinear or noncollinear magnetic configuration, is given in the eighth line. The ninth line gives the Fermi level. From the tenth line, the block data starts. The outer most loop is spin component. The next loop is for $\mathbf{R}$ and the last two are loops of $m$ and $n$, respectively. Each $\mathbf{R}$ is written at the first line of each block together with its degeneracy. The index of $m$ and $n$ is printed and followed by the real and imaginary parts of hopping integrals in each line. An example file, generated by the input file 'Si.dat', is shown here:

```
Real-space Hamiltonian in Wannier Gauge on Wigner-Seitz supercell.
Number of Wannier Function 8
Number of Wigner-Seitz supercell 617
Lattice vector (in Bohr)
   5.10000    0.00000    5.10000
   0.00000    5.10000    5.10000
   5.10000    5.10000    0.00000
collinear calculation spinsize 1
Fermi level -0.112747
R (   -6    2    2 )    4
   1     1     -0.000078903162   -0.000000003750
   1     2      0.000024237763   -0.000000000148
   1     3      0.000024237691   -0.000000000341
   1     4      0.000024238375    0.000000004117
   1     5      0.000072656918   -0.000000000196
   1     6     -0.000022470544   -0.000000000859
   1     7     -0.000022481557    0.000000000750
   1     8     -0.000022492706    0.000000000148
   2     1      0.000024238091    0.000000000049
   2     2     -0.000078901874   -0.000000000011
   2     3      0.000024234912   -0.000000000023
    ........
    .....
    ...
    .
```

### 44.6 Automatic running test of MLWF

To check whether the MLWF calculation part is properly installed or not, an automatic running test for the MLWF calculation can be performed by

#### For the MPI parallel running

```
% mpirun -np 16 openmx -runtestWF
```

#### For the MPI/OpenMP parallel running

```
% mpirun -np 8 openmx -runtestWF -nt 2
```

Then, OpenMX will run with eight test cases, and compare calculated results with the reference results which are stored in 'work/wf_example'. The comparison (absolute difference in the spread and $\Omega$ functions) is stored in a file 'runtestWF.result' in the directory 'work'. The reference results were calculated using a Xeon cluster machine. If the difference is within last seven digits, we may consider that the installation is successful.

# 45 Interface with Wannier90

OpenMX is interfaced with Wannier90 [145] which constructs maximally localized Wannier functions, and calculates physical properties such as Wannier projected DOS and bandstructure, Fermi surface, Berry phase related properties (anomalous Hall conductivity and optical conductivity), and thermoelectric properties. For the calculations, you need to set two keywords as follows:

```
Wannier.Func.Calc        on    # on|off, default=off
Wannier90.fileout        on    # on|off, default=off
```

Once you run a job by an input file with the parameter settings shown above, you will see that the job will finish with the following message:

```
The input files for Wannier90,

System.Name.amn
System.Name.mmn
System.Name.eig
System.Name.win

are successfully generated.
```

After finishing the calculation, you will obtain the four files listed above. The first three files will be read by 'wannier90.x' of Wannier90, and the last file 'System.Name.win' is an input file to handle the calculation of 'wannier90.x'. The schematic computational flow is shown in Fig. 48. With the overlap matrix 'System.Name.mmn', the projection matrix 'System.Name.amn', and eigenvalues 'System.Name.eig', maximally localized Wannier functions are calculated by using 'wannier90.x'. After getting the maximally localized Wannier functions, by using 'postw90.x' of Wannier90 you can calculate a variety of physical properties such as Berry phase related properties (anomalous Hall conductivity and optical conductivity) and thermoelectric properties. In the file 'System.Name.win' the default setting is for the calculation of optical conductivity as

```
berry_task   kubo
```

By changing the option for the keyword 'berry_task' properly, you may be able to calculate other physical quantities. Some of the options for the keyword are listed below:

```
berry_task   kubo     #  optical conductivity
berry_task   ahc      #  anomalous Hall conductivity
```

As for the more details of Wannier90, please refer to the website of Wannier90 [145].

As examples, Figures 49 and 50 show Seebeck coefficient of silicon in the diamond structure and optical conductivity of $SrVO_3$, respectively, calculated by interfacing OpenMX with Wannier90 with the above mentioned scheme. The input files 'Si-Wannier90.dat' and 'SrVO3-Wannier90.dat' can be found in the directory 'work'.

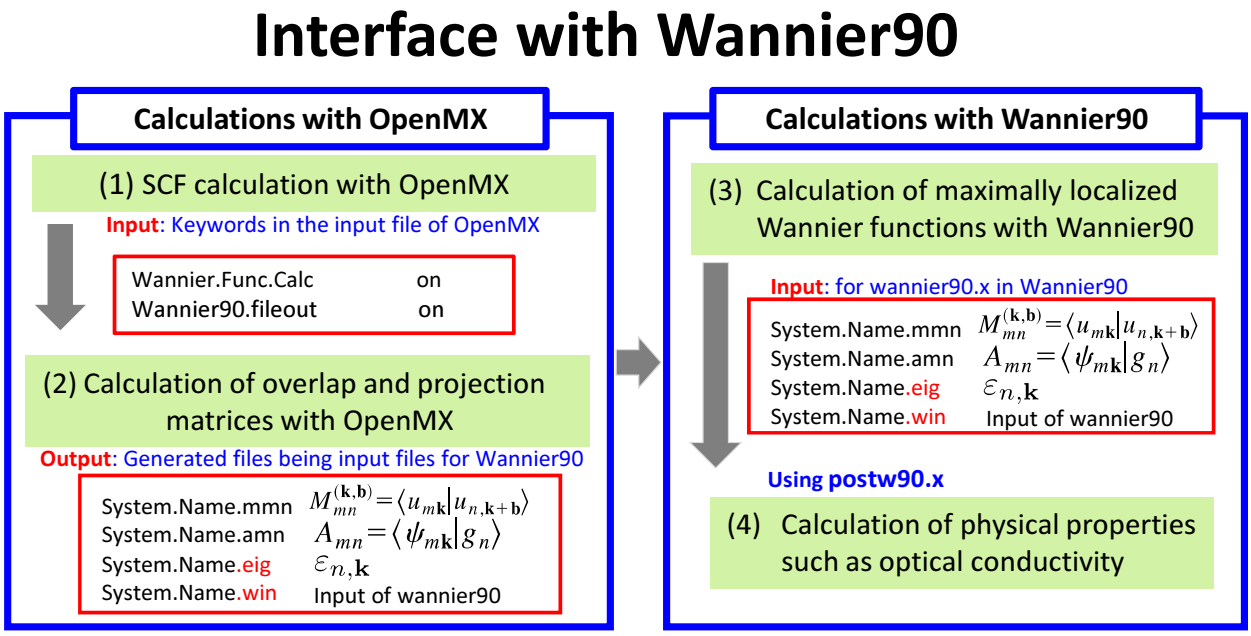



Figure 48: Schematic computational flow for the interface of OpenMX with Wannier90.

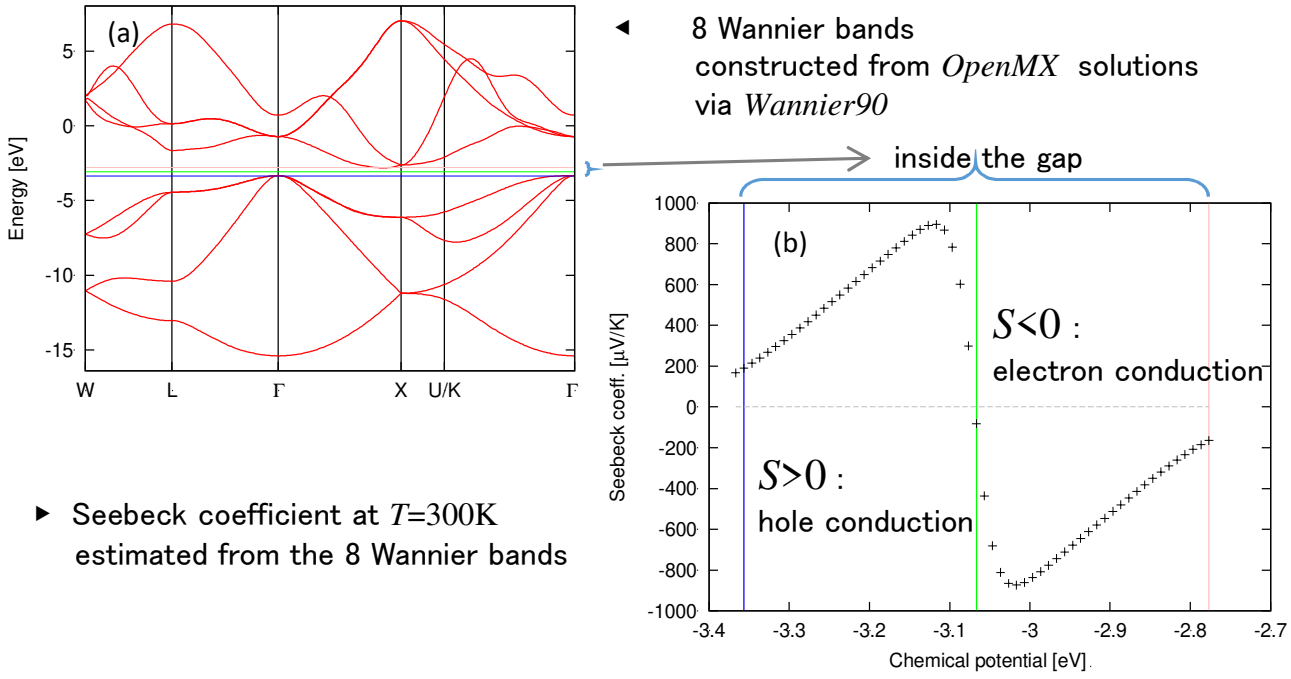



Figure 49: (a) Band structure of silicon in the diamond structure, calculated by 8 Wannier orbitals constructed from OpenMX and Wannier90 calculations, and (b) Seebeck coefficient at T=300K estimated from the 8 Wannier bands. The input file used for the OpenMX calculation is 'Si-Wannier90.dat' which is found in the directory 'work'.

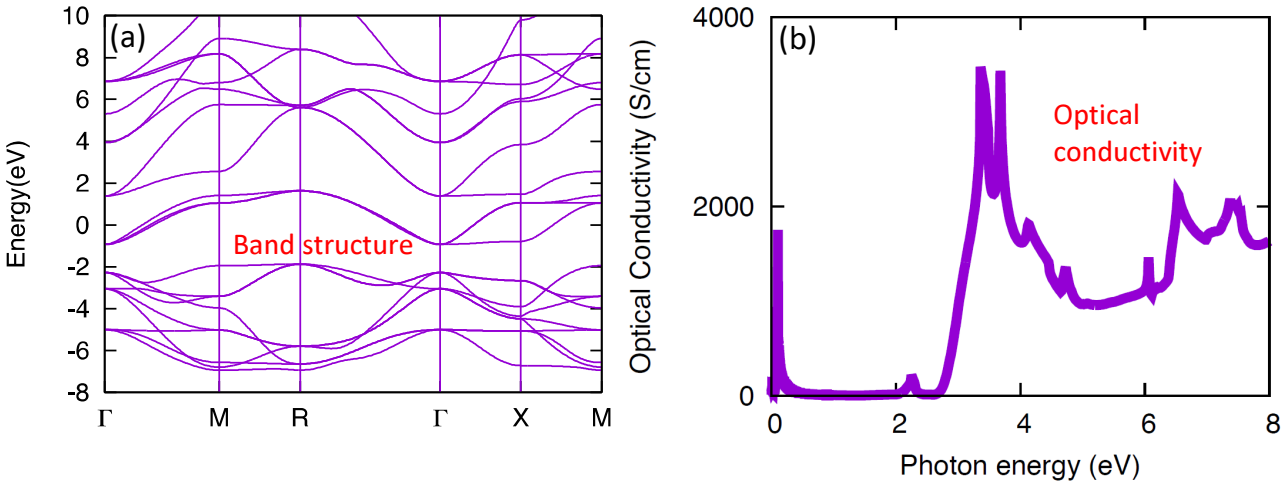



Figure 50: (a) Band structure of $SrVO_3$, (b) optical conductivity of $SrVO_3$ calculated by interfacing OpenMX with Wannier90, which is well compared to an experimental optical conductivity in Ref. [147]. The input file used for the OpenMX calculation is 'SrVO3-Wannier90.dat' which is found in the directory 'work'.

## 46 Numerically exact low-order scaling method for diagonalization

A numerically exact low-order scaling method is supported for large-scale calculations [124]. The computational effort of the method scales as $O(N(\log N)^2)$, $O(N^2)$, and $O(N^{7/3})$ for one, two, and three dimensional systems, respectively, where $N$ is the number of basis functions. Unlike $O(N)$ methods developed so far the approach is a numerically exact alternative to conventional $O(N^3)$ diagonalization schemes in spite of the low-order scaling, and can be applicable to not only insulating but also metallic systems in a single framework. The well separated data structure is suitable for the massively parallel computation as shown in Fig. 47. However, the advantage of the method can be obtained only when a large number of CPU cores are used for parallelization, since the prefactor of computational efforts can be large. When you calculate low-dimensional large-scale systems using a large number of CPU cores, the method can be a proper choice. To choose the method for diagonzalization, you can specify the keyword 'scf.EigenvalueSolver' as

```
scf.EigenvalueSolver      cluster2
```

The method is supported only for colliear DFT calculations of cluster systems or periodic systems with the Γ point for the Brillouin zone sampling. As well as the total energy calculation, the force

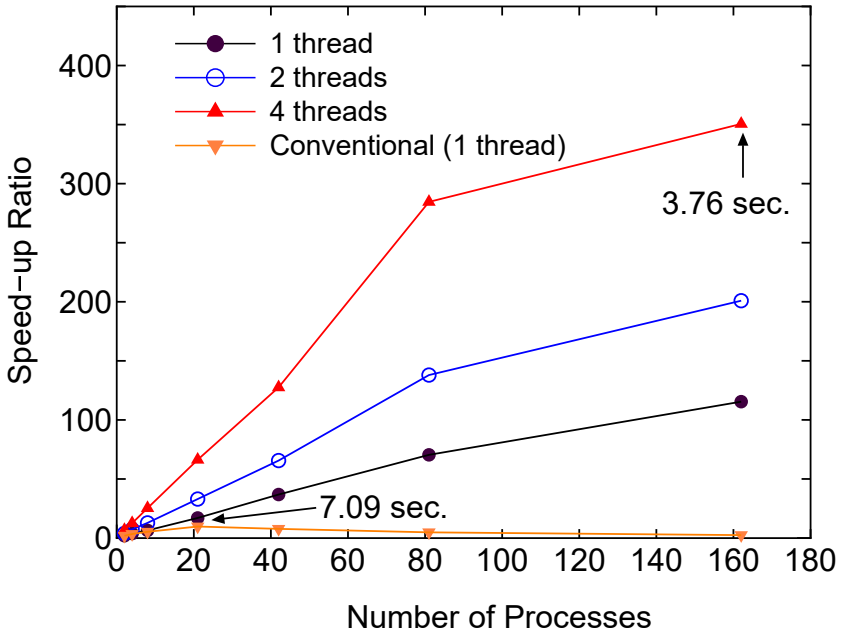



Figure 51: Speed-up ratio in the parallel computation of the diagonalization in the SCF calculation for DNA by a hybrid scheme using MPI and OpenMP. The speed-up ratio is defined by $2T_2/T_p$, where $T_2$ and $T_p$ are the elapsed times obtained by two MPI processes and by the corresponding number of processes and threads. The parallel calculations were performed on a CRAY-XT5 machine consisting of AMD opteron quad core processors (2.3 GHz). The electric temperature of 700 K and 80 poles for the contour integration are used. For comparison, the speed-up ratio for the parallel computation of the conventional scheme using Householder and QR methods is also shown for the case with a single thread. The elapsed time at cases pointed by arrow is also shown for both the low-order scaling and conventional methods.

Table 10: Total energy of a C60 molecule calculated by the numerically exact low-order scaling method and conventional method, and its computational time (sec.) for the diagonalization using 8 processes in the MPI parallelization. The input file is 'C60_LO.dat' in the directory 'work'.

| Method | Total energy (Hartree) | Computational time (sec.) |
|---|---|---|
| Low-order | -343.896238929370 | 69.759 |
| Conventional | -343.896238929326 | 2.784 |

calculation by the low-order scaling method is supported. Thus, it is possible to perform geometry optimization. However, calculations of density of states and wave functions are not supported yet. The number of poles in the contour integration [74] is controlled by a keyword:

```
scf.Npoles.ON2              90
```

The number of poles to achieve convergence does not depend on the size of system [124], but depends on the spectrum radius of system. If the electronic temperature more 300 K is used, the use of 100 poles is enough to get sufficient convergence for the total energy and forces. As an illustration, we show a calculation by the numerically exact low-order scaling method using an input file 'C60_LO.dat' stored in the directory 'work'.

```
% mpirun -np 8 openmx C60_LO.dat
```

As shown in Table 10, the total energy by the low-order scaling method is equivalent to that by the conventional method within double precision, while the computational time is much longer than that of the conventional method for such a small system. We expect that the crossing point between the low-order scaling and the conventional methods with respect to computational time is located at around 300 atoms when using more than 100 cores for the parallel computation, although it depends on the dimensionality of system.

# 47 Effective screening medium method

## 47.1 General

The effective screening medium (ESM) method is a first-principles computational method for charged or biased systems consisting of a slab [125, 126, 127, 128]. In this method, a 2-dimensional periodic and 1-dimensional optional boundary conditions are imposed on a model cell (Fig. 52(a)), and the Poisson's equation is solved under those set of boundary conditions by using the Green's function method. An isolated slab, charged slab, and a slab under a uniform electric field can be treated by introducing the following combinations of semi-infinite media (ESMs).

(a) Isolated slab: vacuum (relative permittivity $\varepsilon = 1$) + vacuum

(b) Charged slab: vacuum + ideal metal (relative permittivity $\varepsilon = \infty$)

(c) Slab under an electric filed: ideal metal + ideal metal

Here *slab* means a system consisting of molecules spaced out 2-dimensionally as well as a slab generally used as a surface model. An isolated slab model can be used for investigations of a polarized substrate, and charged slab model is applicable to a simulation of an electrode surface. A slab model under an electric filed sandwiched between two ideal-metal media would be appropriate for a material located in a metal capacitor. In OpenMX, a unit cell used in an ESM-method calculation is constructed as follows (see Fig. 52(a)):

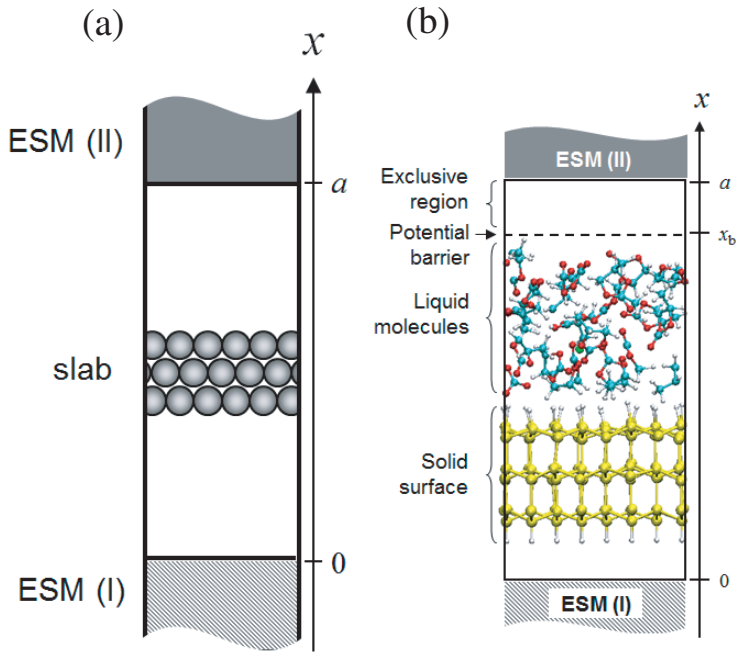



Figure 52: (a) Schematic view of a slab with semi-infinite media (ESMs). ESM (I) and (II) are placed at cell-boundaries, $x = 0$ and $a$ ($a$: the length of the cell along $x$-axis), respectively. (b) An example of a unit cell for a MD calculation of solid surface-liquid interface model system with the ESM method. The slab and ESMs are placed parallel to the $y$-$z$ plane.

1. The **a**-axis of the cell is perpendicular to the **b**-**c** plane and is parallel to the x-axis.
2. Two periodic boundary conditions are set in $y$- and $z$-axis directions
3. ESMs are placed at the cell-boundaries ($x = 0$ and $a$).
4. The origin of the $x$-axis is set at the cell boundary.
5. A fractional coordinate for x-axis is designated between 0 and 1.

A calculation based on the ESM-method can be performed by the following keyword:

```
ESM.switch              on3      # off, on1=v|v|v, on2=m|v|m, on3=v|v|m, on4=on2+EF
ESM.buffer.range        4.5      # default=10.0 (ang),
```

where on1, on2, on3, and on4 represent combinations of ESMs, 'vacuum + vacuum', 'ideal metal + ideal metal', 'vacuum + ideal metal', and 'ideal metal + ideal metal under an electric field', respectively. The keyword 'ESM.buffer.range' indicates the width of an exclusive region for atoms with ESM (unit is Å), which is necessary in order to prevent overlaps between wave functions and ESM.

1. ESM.switch = on1:

   Both ESM (I) and (II) are semi-infinite vacuum media. In this case, note that the total charge of a calculation system should be neutral. The keyword 'scf.system.charge' should be set to zero.

2. ESM.switch = on2:

   Both ESM (I) and (II) are semi-infinite ideal-metal media. One can deal with charged systems. The keyword 'scf.system.charge' can be set to a finite value.

3. ESM.switch = on3:

   ESM (I) and (II) are a semi-infinite vacuum and ideal metal medium, respectively. One can deal with charged systems. The keyword 'scf.system.charge' can be set to be a finite value.

4. ESM.switch = on4:

   An electric field is imposed on the system with the same combination of ESMs to 'on2'. By using the following keyword, one can impose a uniform electric field on a calculation system;

   ```
   ESM.potential.diff        1.0      # default=0.0 (eV),
   ```

   where you can specify a potential difference between two semi-infinite ideal-metal media with reference to the bottom ideal metal (unit is eV). The electric filed is determined by the length of the cell, a, and the potential difference.

5. In case of MD calculations with the ESM method:

   One can perform MD calculations of solid surface-liquid interface systems with any combinations of ESMs. A surface-model slab and a liquid region should be located as shown in Fig. 52(b). In order to restrict liquid molecules within a given region, an cubic barrier potential can be introduced by using the following keyword (see Fig. 52(b)):

```
ESM.wall.position        6.0      # default=10.0 (ang)
ESM.wall.height        100.0      # default=100.0 (eV),
```

where 'ESM.wall.position' denotes the distance between the upper edge of the cell and the origin of the barrier potential, $a - x_b$, and 'ESM.wall.height' is the height of the potential (value of potential energy) at $x = x_b + 1.0$ (Å). It is also recommended to fix positions of atoms on the bottom of a surface-model slab during the MD run.

6. Choice of the axis to be treated by ESM method

   As shown in Fig. 52(a), we assume in the manual that the treatment by the ESM method is applied to the $x$-axis. However, the choice of the axis to be treated by ESM method can be changed by the keyword 'ESM.direction' as

   ```
   ESM.direction            x          # x|y|z, default=x
   ```

   The default direction is the $x$-axis as shown in Fig. 52(a).

### 47.2 Example of test calculation

Let us show effects of ESMs on the electronic structure of a system. As a demonstration calculation, the distribution of excess charge $\rho_{\rm ex}$ in a 1×1 Al-terminated Si(111) slab under the boundary condition, 'vacuum + ideal metal' (ESM.switch = on3), is presented in Fig. 53(a) (the input file of this test calculation 'Al-Si111_ESM.dat' is found in the directory 'work'). It can be seen that segregation of the doped charge in the slab happened due to the attractive interaction between the doped and the corresponding mirror charges. Figure 53(b) indicates the change of the Hartree potential $\Delta V_{\rm H}$ corresponding to each condition as shown in Fig. 53(a), where the potential inside the Al-Si(111) slab and the electric field between the slab and the ideal-metal medium change according to the amount of the doped charge.

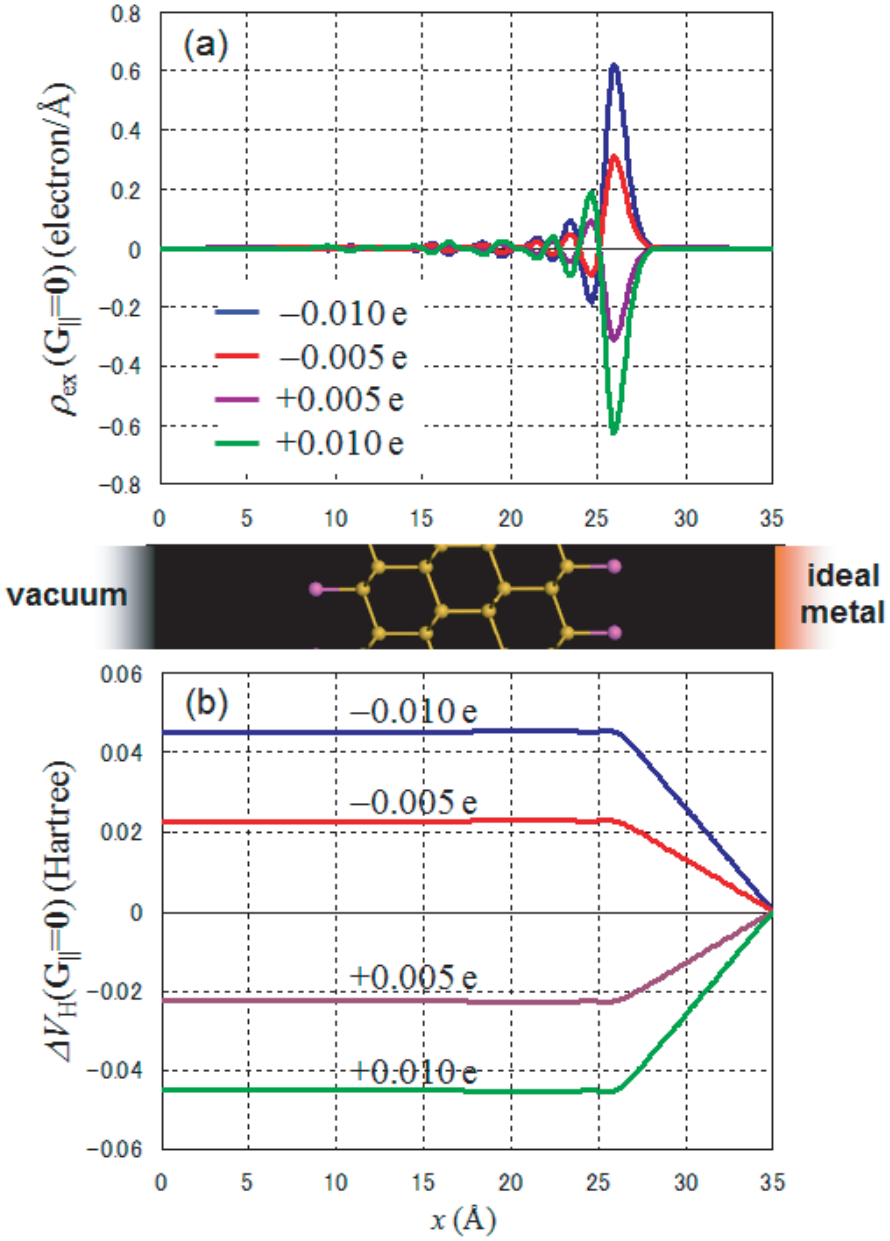



Figure 53: Al-Si(111) slab model with vacuum and ideal-metal ESMs; (a) Distributions of excess charge in Al-Si(111) slab, $\rho_{ex}$; (b) Bias-induced changes of Hartree potentials of Al-Si(111) slab, $\Delta V_H$. The number of doped charge is -0.01, -0.005, +0.005, and +0.01 e. Each plot is obtained as a difference in difference charge or difference Hartree potential with reference to a neutral slab with the same ESMs.

# 48 Calculations of work functions

Using cube files of either *System.Name*.v0.cube or *System.Name*.v1.cube, one can calculate work functions of metals. The work function is defined by

$$\begin{aligned} \Phi &= (\phi_\infty + E[N-1]) - E[N], \\ &= \phi_\infty - \mu, \end{aligned} \tag{11}$$

where $E[N]$ and $E[N-1]$ are the total energies of the $N$-electron and $(N-1)$-electron systems, respectively, and $\phi_\infty$ and $\mu$ are the potential at the infinite distance from the surface and the chemical potential, respectively. The second line of Eq. (11) can be obtained by the Janak theorem [89] as $E[N-1] - E[N] = \int dn \partial E/\partial n = -\mu$, where $n$ is an occupation number of an one-particle eigenstate on the Fermi surface, $dn = -ds/S$ defined with the area of the Fermi surface $S$ and an infinitesimal area $ds$, and the surface integral is performed over the Fermi surface. Since the work function is a quantity associated to a surface, we need to introduce a slab model as shown in Fig. 54(a). As an example, we consider an aluminum slab, where the layer thickness is 5 and the vacuum of about 60 Å is taken into account along the $x$-axis together with the effective screening medium (ESM) method in Sec. 47, where 'ESM.switch=on1' is used, to avoid the interaction between the periodic slabs. The SCF calculation can be performed using an input file 'Al111_WorkFunc_0E.dat' as follows:

```
% mpirun -np 28 ./openmx Al111_WorkFunc_0E.dat
```

After the calculation is completed normally, you obtain a cube file of 'Al111_WorkFunc_0E.v0.cube'. To analyze the cube file, you can utilize a post-processing code of 'gcube2oned.c' in the directory 'source', which can be compiled as

```
% gcc gcube2oned.c -lm -o gcube2oned
```

After copying the executable code 'gcube2oned' to your working directory, you can transform the data of 3D cube data to an 1D data along a chosen direction, in this case '1' corresponding to the **a**-axis, by integrating over the remaining 2D (**bc**-plane) as follows:

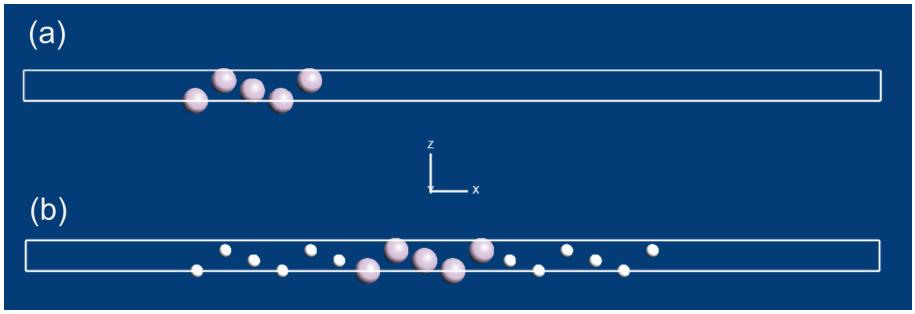



Figure 54: (a) A slab model consisting of Al atoms, where the layer thickness is 5 and the vacuum of 60 Å is taken into account along the $x$-axis. (b) A slab model consisting of Al atoms, where the layer thickness is 5 and the vacuum of 60 Å is taken into account along the $x$-axis. In addtion to the Al layers, the 6 layers consisting of empty atoms are introduced for both the surfaces so that the tail of wave function towards the vacuum region can be accurately described.

```
% gcube2oned Al111_WorkFunc_0E.v0.cube 1 > 1d_pot.txt
```

In the obtained file '1d_pot.txt', the first, second, and third columns correspond to the serian number of grid, the position (Å) along the **a**-axis, and a 1D potential (Hartree) averaged over the **bc**-plane. In case of the 1D potential along the **b** or **c**-axis, you can specify '2' or '3' as an argument of 'gcube2oned'. In Fig. 55 we show the 1D potentials for the Al(111) surface along the **a**-axis which is perpendicular to the surface. The number of layers consisting of empty atoms is systematically changed from 0 to 7 for both the surfaces. As shown in Fig. 54, the layers consisting of empty atoms are taken into account so that the tail of wave function towards the vacuum region can be accurately described. The input files used for the calculations are 'Al111_WorkFunc_%E.dat', where % varies from 0 to 7, which are all available in the directory 'work'. The potential $\phi_\infty$ in Eq. (11) can be obtained from that at around 70 Å, while the chemical potential $\mu$ can be found from the out file. Since the potential at around 70 Å is almost zero, the work function is basically determined by the chemical potential in the cases. Then, the calculated values using Eq. (11) are plotted as a function of the number of empty layers in Fig. 56. One can see that the work function of the Al(111) surface reaches to the convergence at the 2 empty layers, implying adding the 2 empty layers is sufficient to obtain the convergent result.

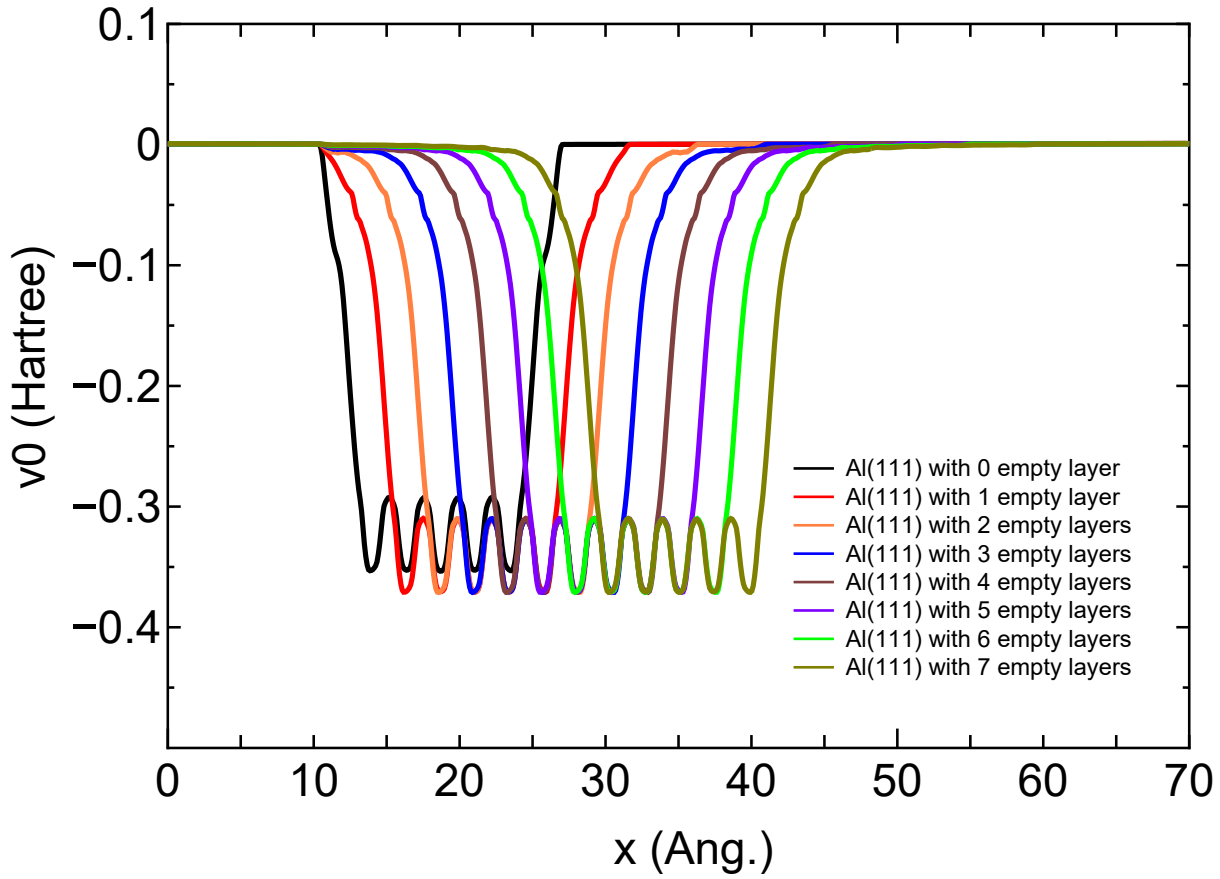



Figure 55: 1D potentials (Hartree) along the **a**-axis averaged over the **bc**-plane, which are obtained from the v0.cube files. The number of layers consisting of empty atoms is systematically changed from 0 to 7. The input files used for the calculations are 'Al111_WorkFunc_%E.dat', where % varies from 0 to 7, which are all available in the directory 'work'.

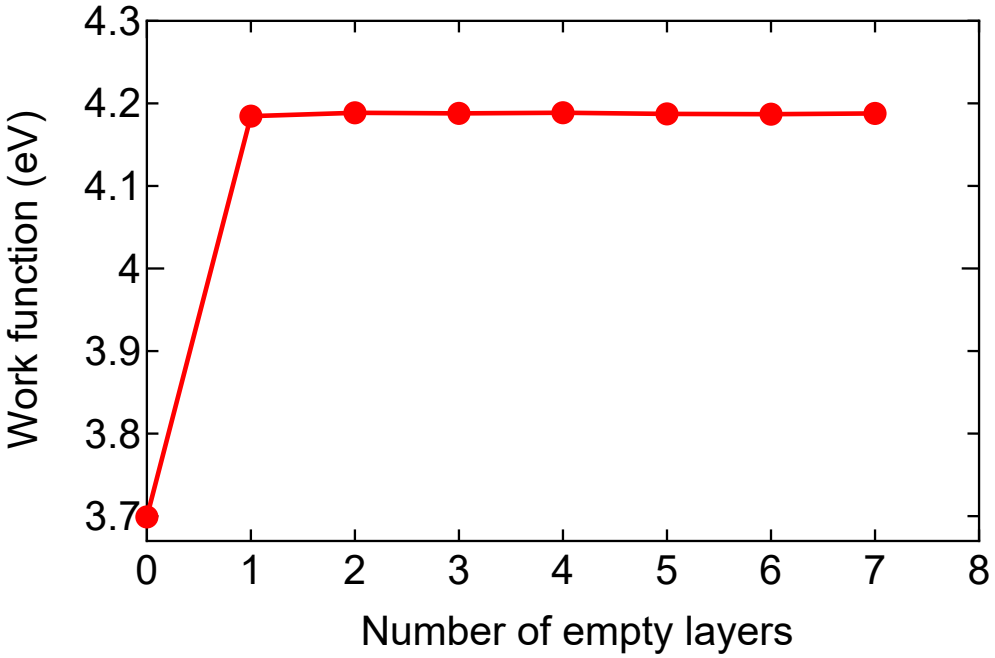



Figure 56: Work function (eV) of the Al(111) surface as a function of the number of empty layers. The values can be obtained based on Eq. (11) and the calculations shown in Fig. 55.

In Table 11 we show the calculated results of work function for five metals and the corresponding experimental values. In all the calculations the 7 empty layers were introduced. It is found that the calculated values are well compared to the experimental values. The input files used for the calculations are 'Al111_WorkFunc_7E.dat', 'Cu111_WorkFunc_7E.dat', 'Ag111_WorkFunc_7E.dat', 'Au111_WorkFunc_7E.dat', and 'Pd111_WorkFunc_7E.dat', which are all available in the directory 'work'.

As for gapped systems Eq. (11) may not be valid in a rigorous sense. However, you may be able to use Eq. (11) as an approximate treatment. Especially for gapped systems with polar surfaces, you need to use the ESM method in Sec. 47 to avoid the interaction between the periodic slabs.

Table 11: Calculated work functions (eV) for five metals and the corresponding experimental values. We note that the reported experimental values seem to vary in literatures, and we show one of them in the table.

| | OpenMX | Expt. |
|---|---|---|
| Al(111) | 4.19 | 4.26±0.03 [129] |
| Cu(111) | 4.74 | 4.94±0.03 [130] |
| Ag(111) | 4.51 | 4.46±0.02 [131] |
| Au(111) | 5.33 | 5.26±0.04 [132] |
| Pd(111) | 5.40 | 5.55±0.01 [133] |

# 49 Nudged elastic band (NEB) method

## 49.1 General

To search a minimum energy path (MEP) in geometrical phase space connecting two stable structures, a nudged elastic band (NEB) method based on Ref. [134] is supported in OpenMX Ver. 3.9. The detail of the implementation is summarized as follows:

- Calculation of tangents based on Eqs. (8)-(11) in Ref. [134]
- Calculation of perpendicular forces based on Eq. (4) in Ref. [134]
- Calculation of parallel forces based on Eq. (12) in Ref. [134]
- Optimization method based on a hybrid DIIS+BFGS optimizer

In order to minimize user's efforts in using it, the functionality of NEB has been realized as one of geometry optimizers with the following features:

- Easy to use
- Hybrid MPI/OpenMP parallelization
- Initial path by the straight line or user's definition
- Only three routines added

## 49.2 How to perform

The NEB calculation is performed by the following three steps:

1. Geometry optimization of a precursor
2. Geometry optimization of a product
3. Optimization of a minimum energy path (MEP) connecting the precursor and product

where in the three calculations users have to keep the same computational parameters such as unit cell, cutoff energy, basis functions, pseudopotentials, and electronic temperatures to avoid numerical inconsistency. After the calculations 1 and 2, files '*System.Name*.dat#' are generated. By using the atomic coordinates in the files '*System.Name*.dat#', one can easily construct an input file for the calculation 3. Once you have an input file for the calculation 3, the execution of the NEB calculation is the same as for the conventional OpenMX calculation such as

```
% mpirun -np 32 openmx input.dat -nt 4
```

## 49.3 Examples and keywords

Two input files are provided as example:

- C2H4_NEB.dat

  Cycloaddition reaction of two ethylene molecules to cyclobutane

- Si8_NEB.dat

  Diffusion of an interstitial hydrogen atom in the diamond Si

The input file 'C2H4_NEB.dat' will be used to illustrate the NEB calculation in the proceeding explanation.

### Providing two terminal structures

The atomic coordinates of the precursor are specified in the input file by

```
<Atoms.SpeciesAndCoordinates
1    C   -0.66829065594143    0.00000000101783   -2.19961193219289     2.0     2.0
2    C    0.66817412917689   -0.00000000316062   -2.19961215251205     2.0     2.0
3    H    1.24159214112072   -0.92942544650857   -2.19953308980064     0.5     0.5
4    H    1.24159212192367    0.92942544733979   -2.19953308820323     0.5     0.5
5    H   -1.24165800644131   -0.92944748269232   -2.19953309891389     0.5     0.5
6    H   -1.24165801380425    0.92944749402510   -2.19953309747076     0.5     0.5
7    C   -0.66829065113509    0.00000000341499    2.19961191775648     2.0     2.0
8    C    0.66817411530651   -0.00000000006073    2.19961215383949     2.0     2.0
9    H    1.24159211310925   -0.92942539308841    2.19953308889301     0.5     0.5
10    H    1.24159212332935    0.92942539212392    2.19953308816332     0.5     0.5
11    H   -1.24165799549343   -0.92944744948986    2.19953310195071     0.5     0.5
12    H   -1.24165801426648    0.92944744880542    2.19953310162389     0.5     0.5
Atoms.SpeciesAndCoordinates>
```

The atomic coordinates of the product are specified in the input file by

```
<NEB.Atoms.SpeciesAndCoordinates
1    C   -0.77755846408657   -0.00000003553856   -0.77730141035137     2.0     2.0
2    C    0.77681707294741   -0.00000002413166   -0.77729608216595     2.0     2.0
3    H    1.23451821718817   -0.88763832172374   -1.23464057728123     0.5     0.5
4    H    1.23451823170776    0.88763828275851   -1.23464059022330     0.5     0.5
5    H   -1.23506432458023   -0.88767426830774   -1.23470899088096     0.5     0.5
6    H   -1.23506425800395    0.88767424658723   -1.23470896874564     0.5     0.5
7    C   -0.77755854665393    0.00000000908006    0.77730136931056     2.0     2.0
8    C    0.77681705017323   -0.00000000970885    0.77729611199476     2.0     2.0
9    H    1.23451826851556   -0.88763828740000    1.23464060936812     0.5     0.5
10    H    1.23451821324627    0.88763830875131    1.23464061208483     0.5     0.5
11    H   -1.23506431230451   -0.88767430754577    1.23470894717613     0.5     0.5
12    H   -1.23506433587007    0.88767428525317    1.23470902573029     0.5     0.5
NEB.Atoms.SpeciesAndCoordinates>
```

### Keywords for the NEB calculation

The NEB calculation can be performed by setting the keyword 'MD.Type' as

```
MD.Type                 NEB
```

The number of images in the path is given by

```
MD.NEB.Number.Images        8           # default=10
```

where the two terminals are excluded from the number of images.
The spring constant is given by

```
MD.NEB.Spring.Const         0.1         # default=0.1(hartee/borh^2)
```

In most cases, the obtained path does not largely depend on the value.
The optimization of MEP is performed by a hybrid DIIS+BFGS scheme which is controlled by the following keywords:

```
MD.Opt.DIIS.History           4         # default=7
MD.Opt.StartDIIS             10         # default=5
MD.maxIter                  100         # default=1
MD.Opt.criterion         1.0e-4         # default=1.0e-4 (Hartree/Bohr)
```

The specification of these keywords are the same as for the geometry optimization. So, see the section 'Geometry optimization' in the manual for the details. Also, it is also possible to fix the atomic position by the keyword 'MD.Fixed.XYZ'.

**Execution of the NEB calculation**

One can perform the NEB calculation with the input file 'C2H4_NEB.dat' by

```
% mpirun np 16 openmx C2H4_NEB.dat
```

If the calculation is successfully completed, more than 24 files will be generated. Some of them are listed below:

```
c2h4.neb.opt          history of optimization for finding MEP
c2h4.neb.ene          total energy of each image
c2h4.neb.xyz          atomic coordinates of each image in XYZ format
C2H4_NEB.dat#         input file for restarting.
C2H4_NBE.dat_0        input file for the precursor
C2H4_NBE.dat_1        input file for the image 1
C2H4_NBE.dat_2        input file for the image 2
C2H4_NBE.dat_3        input file for the image 3
C2H4_NBE.dat_4        input file for the image 4
C2H4_NBE.dat_5        input file for the image 5
C2H4_NBE.dat_6        input file for the image 6
C2H4_NBE.dat_7        input file for the image 7
C2H4_NBE.dat_8        input file for the image 8
C2H4_NBE.dat_9        input file for the product
c2h4_0.out            output file for the precursor
c2h4_1.out            output file for the image 1
```

```
c2h4_2.out            output file for the image 2
c2h4_3.out            output file for the image 3
c2h4_4.out            output file for the image 4
c2h4_5.out            output file for the image 5
c2h4_6.out            output file for the image 6
c2h4_7.out            output file for the image 7
c2h4_8.out            output file for the image 8
c2h4_9.out            output file for the product
```

'c2h4.neb.opt' contains history of optimization for finding MEP as shown in Fig. 53(a). One can see the details at the header of the file as follows:

```
***********************************************************
***********************************************************
History of optimization by the NEB method
***********************************************************
***********************************************************

iter   SD_scaling     |Maximum force|   Maximum step        Norm        Sum of Total Energy of Images
(Hartree/Bohr)        (Ang)       (Hartree/Bohr)       (Hartree)


1       0.37794520       0.12552539       0.04583072       0.49503563      -223.77727271
2       0.37794520       0.08684953       0.03163814       0.35379139      -223.85742175
3       0.37794520       0.05494411       0.01922344       0.25668987      -223.89831309
4       0.37794520       0.03790970       0.01234783       0.20282699      -223.92042217
5       0.45353424       0.02936250       0.01326992       0.17349184      -223.93482686
6       0.45353424       0.02588308       0.01169327       0.15249816      -223.94772371
7       0.45353424       0.02303223       0.01039732       0.13836350      -223.95785384
.....
...
.
```

Also, 'c2h4.neb.ene' and 'c2h4.neb.xyz' can be used to analyze the change of total energy as a function of the distance (Bohr) from the precursor and the structural change as shown in Fig. 57(b). The content of 'c2h4.neb.ene' is as follows:

```
#
# 1st column: index of images, where 0 and MD.NEB.Number.Images+1 are the terminals
# 2nd column: Total energy (Hartree) of each image
# 3rd column: distance (Bohr) between neighbors
# 4th column: distance (Bohr) from the image of the index 0
#
  0     -28.02185123       0.00000000       0.00000000       1.33646479
  1     -28.02178507       0.82118927       0.82118927       1.33567761
  2     -28.02140083       0.82112464       1.64231391       1.33523542
  3     -28.02029258       0.82111520       2.46342911       1.33463918
  4     -28.01779519       0.82113225       3.28456136       1.33375873
  5     -28.01261498       0.82135735       4.10591871       1.33262670
  6     -27.98761576       0.82169347       4.92761218       1.34184319
```

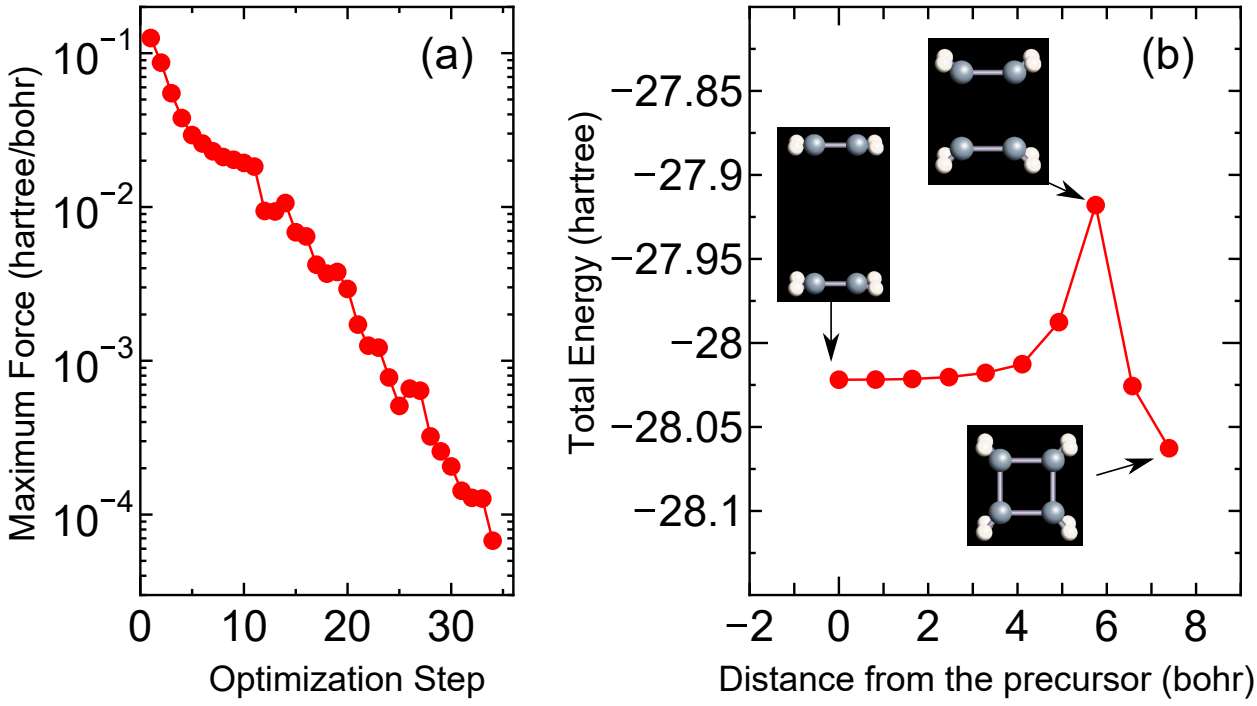



Figure 57: (a) History of optimization (c2h4.neb.opt) for the NEB calculation for a cycloaddition reaction of two ethylene molecules to a cyclobutane molecule, (b) change of total energy (c2h4.neb.ene) of two ethylene molecules as a function of the distance (Bohr) from the precursor and the corresponding geometrical structures (c2h4.neb.xyz) of images on the minimum energy path. The input file used for the NEB calculation is 'C2H4_NEB.dat' in the directory 'work'.

```
7     -27.91797754     0.82218705     5.74979923     1.51281867
8     -28.02565242     0.82256542     6.57236464     1.55582513
9     -28.06263668     0.82263897     7.39500361     1.55437554
```

where the first column is a serial number of image, while 0 and 9 correspond to the precursor and product, respectively. The second column is the total energy of each image. The third and fourth columns are interval (Bohr) between two neighboring images and the distance (Bohr) from the precursor in geometrical phase space. A file '*System.Name*.dat_#', where '*System.Name*' is 'System.Name' and '#' is a serial number for each image, is also generated, since each calculation for each image is basically done as an independent OpenMX calculation with a different input file. A corresponding output file '*System.Name*_#.out' is also generated, which may be useful to analyze how the electronic structure changes on MEP.

As well as the case of 'C2H4_NEB.dat', one can perform the NEB calculation by 'Si8_NEB.dat'. After the successful calculation, you may get the history of optimization and change of total energy along MEP as shown in Fig. 58.

### 49.4 Restarting the NEB calculation

It often happens that the convergence is not achieved even after the maximum optimization step. In such a case, one has to continue the optimization as a new job starting from the last optimization step

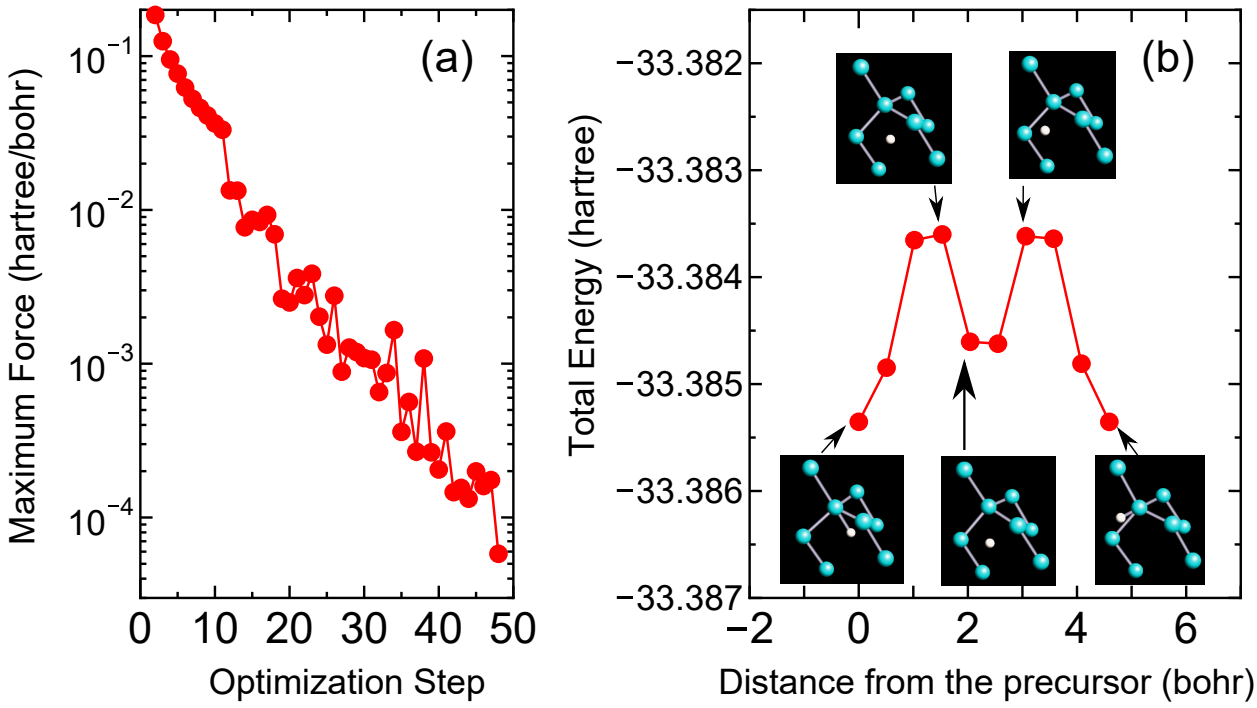



Figure 58: (a) History of optimization (si8_neb.neb.opt) for the NEB calculation for diffusion of an interstitial hydrogen atom in the diamond Si, (b) change of total energy (si8_neb.neb.ene) as a function of the distance (Bohr) from the precursor and the corresponding geometrical structures (si8_neb.neb.xyz) of images on the minimum energy path. The input file used for the NEB calculation is 'Si8_NEB.dat' in the directory 'work'.

in the previous job. A file '*System.Name*.dat#' is generated after every optimization step. The file contains a series of atomic coordinates for images in the last step. One can restart the optimization using a file '*System.Name*.dat#'.

### 49.5 User defined initial path

As default, the initial path connecting the precursor and the product is a straight line connecting them. However, in some cases the geometrical structure of images generated on the straight line can be very erratic so that distance between atoms can be too close to each other. In this case, one should explicitly provide the atomic coordinates of images. The user defined initial path can be provided by the same way as for the restarting. Then, one has to provide atomic coordinates for each image by the following keywords:

```
<NEB1.Atoms.SpeciesAndCoordinates
1   Si   -0.12960866043083    0.13490502997627   -0.12924862991035     2.0     2.0
2   Si   -0.40252421446808    5.19664433048606    4.91248322056082     2.0     2.0
...
NEB1.Atoms.SpeciesAndCoordinates>

<NEB2.Atoms.SpeciesAndCoordinates
1   Si   -0.08436294149342   -0.02173837971883   -0.08374099211565     2.0     2.0
```

```
2   Si   -0.33677725120015    5.10216241168093    5.01087499461541     2.0     2.0
...
NEB2.Atoms.SpeciesAndCoordinates>
```

For all the images of which number is given by 'MD.NEB.Number.Images', the atomic coordinates need to be provided. Also, it is required for a keyword to be switched on as

```
scf.restart     on
```

## 49.6 Monitoring the NEB calculation

In the NEB calculation, the standard output will display only that for the image 1, and those for the other images will not be displayed. However, there is no guarantee that the SCF iteration converges for all the images. In order to monitor the SCF convergence for all the images, temporary files can be checked by users. In the NEB calculation, an input file is generated for each image, whose name is '*System.Name*.dat_#', where '#' runs from 0 to MD.NEB.Number.Images+1, and 'system.name' is modified as the original system.name_#. So, one can check the SCF convergence by monitoring a file 'system.name_#.DFTSCF', whether it converges or not.

## 49.7 Parallel calculation

In the NEB calculation, the setting for the parallelization will be automatically done depending on the number of processes and threads. However, it would be better to provide a proper number of processes for the MPI parallelization which can be divisible by the number of images given by 'MD.NEB.Number.Images', in order to achieve a good load balance in the MPI parallelization. It is noted that the number of processes for the MPI parallelization can exceed the number of atoms. The hybrid parallelization by MPI/OpenMP is also supported.

Although the default parallelization scheme works well in most cases, a memory shortage can be a serious problem when a small number of the MPI processes is used for large-scale systems. In the default MPI parallelization, the images are preferentially parallelized at first. When the number of MPI processes exceeds the number of images, the calculation of each image starts to be parallelized, where the memory usage starts to be parallelized as well. In this case, users may encounter a segmentation fault due to the memory shortage if many CPU cores are not available. To avoid such a situation, the following keyword is available.

```
MD.NEB.Parallel.Number    3
```

In this example, the calculations of every three images are parallelized at once where the MPI processes are classified to three groups and utilized for the parallelization of each image among the three images. In order to complete the calculations of all the images, the grouped calculations are repeated by floor[(the number of images)/(MD.NEB.Parallel.Number)] times. The scheme may be useful for the NEB calculation of a large-scale system. If the keyword is not specified in your input file, the default parallelization scheme is employed.

## 49.8 Other tips

It would be better to provide atomic coordinates for bulk systems in Ang or AU instead of FRAC, since the atomic position tends to be translated in FRAC to keep the fractional coordinate within 0 to 1. The translation tends to generate a confusing movie in the visualization of the result.

Only three routines are added to implement the NEB functionality. They are 'neb.c', 'neb_run.c', and 'neb_check.c'. The main routine is 'neb.c'. It may be easy to implement related methods in 'neb.c'.

## 50 STM image by the Tersoff-Hamann scheme

Scanning tunneling microscope (STM) image can be obtained by the Tersoff-Hamann scheme [71]. The method is nothing but calculation of partial charge density in an energy window measured from the chemical potential. The calculation of the partial charge density is performed by the following keywords:

```
partial.charge                  on       # on|off, default=off
partial.charge.energy.window    0.0      # in eV
```

where the second keyword defines an energy window (in eV) measured from the chemical potential (a plus value means conduction band and negative valence). Since the calculation of the partial charge density is performed during calculation of the density of states (DOS), the following keywords have to be specified as well:

```
Dos.fileout                    on        # on|off, default=off
Dos.Erange               -20.0  20.0    # default = -20 20
Dos.Kgrid                   5 5 5       # default = Kgrid1 Kgrid2 Kgrid3
```

After the calculation with the keywords, you will get '*System.Name*.pden.cube' which can be used for the STM simulation within the Tersoff-Hamman approximation. As an example, a simulated STM image of a graphene layer is shown in Fig. 59.

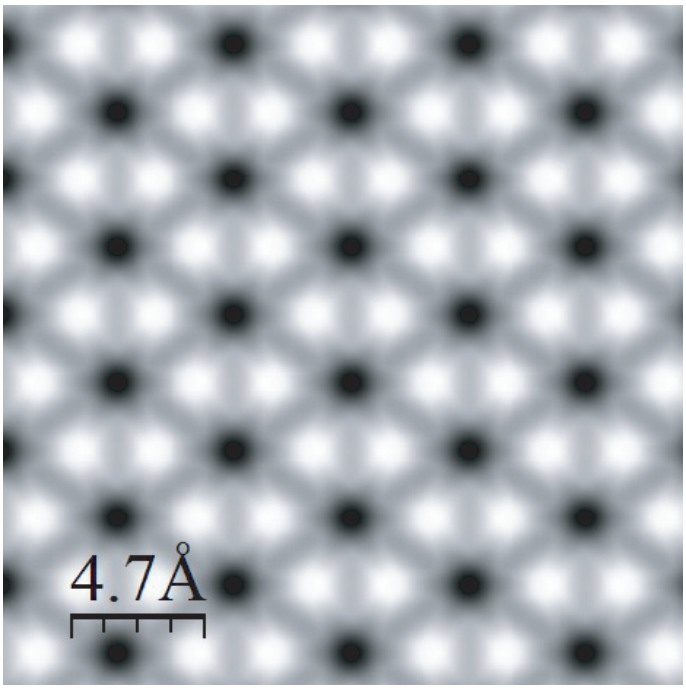



Figure 59: Simulated STM image of a graphene layer, where 'partial.charge.energy.window' of 2 eV was used in the calculation, and the input file is 'Graphene_STM.dat' in the directory 'work'. The cube file 'Graphene_STM.pden.cube' was visualized with an isovalue of 0.0001 by a software WSxM [153].

# 51 DFT-D2 and DFT-D3 for vdW interaction

The DFT-D2 method [135] and the DFT-D3 method [136, 137] by Grimme et al. are supported to include a vdW interaction. In the following subsections we will explain how to use them.

## 51.1 DFT-D2 method

The following keywords are relevant to the DFT-D2 method.

```
scf.dftD                    on          # on|off, default=off
DFTD.Unit                  Ang          # Ang|AU
DFTD.rcut_dftD           100.0          # default=100 (DFTD.Unit)
DFTD.d                    20.0          # default=20
DFTD.scale6               0.75          # default=0.75
DFTD.IntDirection        1 1 1          # default=1 1 1 (1:on 0:off)
```

When you include the vdW correction, switch on 'scf.dftD'. The cutoff radius for the pairwise interaction is given by 'DFTD.rcut_dftD', where the unit is given by 'DFTD.Unit'. The 'd' value in Eq. (12) in Grimme's paper [135] is given by 'DFTD.d', while the default value is 20. The scaling factor in Eq. (11) in Grimme's papar [135] is given by 'DFTD.scale6', while the default value for the PBE functional is 0.75. Also, the interaction can be cut along the **a**-, **b**-, and **c**-axes by 'DFTD.IntDirection', where 1 means that the interaction is included, and 0 not. Also, the periodicity for each atom can be controlled by

```
<DFTD.periodicity
1   1
2   1
3   1
4   1
....
DFTD.periodicity>
```

where the first column is a serial number which is the same as in the 'Atoms.SpeciesAndCoordinates', and the second column is a flag which means that 1 is periodic, and 0 is non-periodic for the corresponding atom. By considering the periodicity or non-periodicity of each atom, the interaction is automatically cut when they are non-periodic.

The main modifications are placed at only two routines: DFTDvdW_init.c and Calc_EdftD() of Total_Energy.c. In DFTDvdW_init.c, you can easily change the parameters for the vdW correction, and in Calc_EdftD() of Total_Energy.c you can confirm how they are calculated.

Since OpenMX uses localized orbitals as basis function, it is very important to take account of basis set superposition error (BSSE) when we investigate an effect of a weak interaction such as vdW interaction. To estimate BSSE, the counterpoise (CP) method [46, 47] can be used. As for the CP method, see the Section 'Empty atom scheme'.

## 51.2 DFT-D3 method

The DFT-D3 method of Grimme et al. [136, 137] is supported to include a vdW interaction with default parameters for the GGA-PBE functional. The following keywords are relevant for the DFT-

D3 method.

```
scf.dftD                    on          # on|off, default=off
version.dftD                 3          # 2|3, default=2
DFTD3.damp                  bj          # zero|bj, default=bj
DFTD.Unit                   AU          # Ang|AU
DFTD.rcut_dftD           100.0          # default=100 (DFTD.Unit)
DFTD.cncut_dftD             40          # default=40 (DFTD.Unit)
DFTD.IntDirection        1 1 1          # default=1 1 1 (1:on 0:off)
```

When you include the DFT-D2 or DFT-D3 calculation, turn on 'scf.dftD'. For DFT-D2 use version.dftD=2 and for DFT-D3 version.dftD=3. The DFT-D3 implemented here supports both zero and Becke-Johnson (BJ) damping functions [137]. The cutoff radius for the interaction is given by 'DFTD.rcut_dftD' and for the coordination number calculation 'DFTD.cncut_dftD'. The units are given by 'DFTD.Unit' and the suggested defaults for both cutoff values are in AU. Also, the interaction for image atoms can be cut along the **a**-, **b**-, and **c**-axes by 'DFTD.IntDirection', where 1 means that the interaction is included, and 0 not. Also, the periodicity for each atom can be controlled as in the case of the DFT-D2 method by

```
<DFTD.periodicity
1   1
2   1
3   1
4   1
....
DFTD.periodicity>
```

where the first column is a serial number which is the same as in the 'Atoms.SpeciesAndCoordinates', and the second column is a flag which means that 1 is periodic, and 0 is non-periodic for the corresponding atom. By considering the periodicity or non-periodicity of each atom, the interaction is automatically cut when they are non-periodic.

The main modifications are placed at only two routines: DFTD3vdW_init.c and Calc_EdftD() of Total_Energy.c. In DFTD3vdW_init.c, you can easily change the parameters for the vdW correction, and in Calc_EdftD3() of Total_Energy.c you can confirm how they are calculated.

Parameters for other functionals may be set through the following keywords:

```
DFTD.scale6              1     # default=0.75|1.0 (for DFT-D2|DFT-D3)
DFTD.scale8         0.7875     # default=0.722|0.7875 (for PBE with zero|bj damping)
DFTD.sr6             1.217     # default=1.217 (for PBE)
DFTD.a1             0.4289     # default=0.4289 (for PBE)
DFTD.a2             4.4407     # default=4.4407 (for PBE)
```

The '$s_6$' and '$s_8$' global scaling value of Eq. (3) in Grimme's paper [136] is given by 'DFTD.scale6' and 'DFTD.scale8'. The global scaling parameters are functional and damping-function dependent. The parameter 'sr6' of Eq. (6) in [136] needs to be set when using the zero damping function while the parameters 'a1' and 'a2' of Eq. (6) in [137] need to be set when choosing BJ damping.

As an example for the DFT-D3 calculation, the interaction energy between two benzene molecules in a parallel structure with $D_{6h}$ symmetry is shown as a function of the inter-distance in Fig. 60. All the input files for the calculations can be found in a directory 'work/DFT-D3/', and they are

```
Dimer-Ben-10.0.dat  Dimer-Ben-3.88.dat  Dimer-Ben-4.5.dat  Mono-Ben-1.dat
Dimer-Ben-3.3.dat   Dimer-Ben-3.89.dat  Dimer-Ben-5.0.dat  Mono-Ben-2.dat
Dimer-Ben-3.4.dat   Dimer-Ben-3.8.dat   Dimer-Ben-6.0.dat  Mono-Ben.dat
Dimer-Ben-3.6.dat   Dimer-Ben-3.9.dat   Dimer-Ben-7.0.dat
Dimer-Ben-3.86.dat  Dimer-Ben-4.0.dat   Dimer-Ben-8.0.dat
Dimer-Ben-3.87.dat  Dimer-Ben-4.2.dat   Dimer-Ben-9.0.dat
```

After optimizing the monomer using 'Mono-Ben.dat', the total energy of dimer in a variety of inter-distance was calculated using 'Dimer-Ben-#.dat' (#=3.3-9.0), where the structure of the benzene molecule is the same as the structure of monomer obtained by the first calculation. The monomer calculations with a counterpoise correction were performed by 'Mono-Ben-1.dat' and 'Mono-Ben-2.dat'. The optimum inter-distance is found to be 3.87 Å, which is well compared with a reported value (3.89 Å) computed with density fitted local second-order Møller-Plesset perturbation theory (DF-LMP2) [138]. The counterpoise corrected interaction energy is 1.73 kcal/mol being in good agreement with a reported value (1.7 kcal/mol) [138], while the basis set superposition error is found to be large.

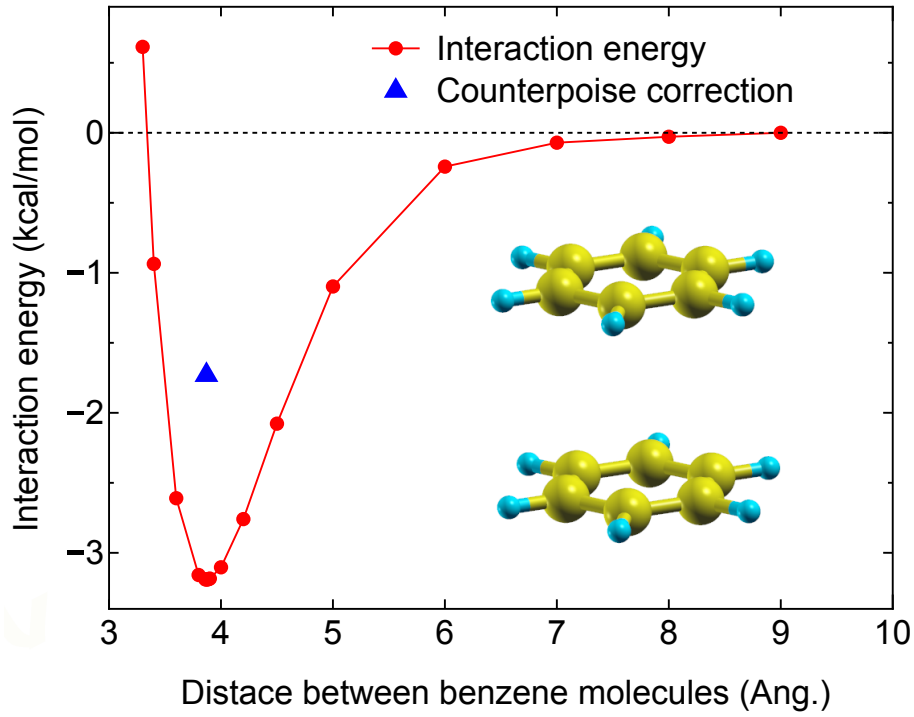



Figure 60: The interaction energy of between two benzene molecules in a parallel structure with $D_{6h}$ symmetry. The counterpoise corrected interaction energy is shown by the triangle. All the input files for the calculations can be found in a directory 'work/DFT-D3/.

# 52 Unfolding method for band structures

When the band structure of a system with imperfection such as surfaces, impurities, vacancies, and structural distortion calculated by the supercell approach is compared to spectrum measured by Angle-Resolved Photoemission Spectroscopy (ARPES), the experimentally measured periodicity of the system is generally different from that of the supercell we introduced for the calculation. In such a case, the band structure obtained by the supercell calculation should be represented for the Brillouin zone of a proper unit cell so that the periodicity of calculated band structure can be consistent with the measured one. Though the choice of the proper unit cell is not obvious in general cases, OpenMX provides a method for unfolding the band structure of the supercell into the Brillouin zone of a reference unit cell that a user specifies in the input file [142]. The functionality is supported for not only collinear, but also non-collinear DFT calculations. Even in case that you are not interested in comparison between calculations and experiments by unfolding the band structrue, the method would be useful to analyze how each band consists of pseudo-atomic orbitals, providing information for physical nature of bands, and how each band is pertubed by the introduced imperfection.

In the following subsequent subsections we explain how these functionalities can be utilized by demonstrating a series of calculations.

## 52.1 Analysis of band structures

First, let us analyze how each band can be decomposed into each contribution of pseudo-atomic orbital in a band structure calculation for a primitive cell of SiC in a two-dimensional honeycomb structure without imperfection. Note that unfolding bands is not performed in this case. The SCF calculation for the primitive cell of the two-dimensional SiC can be performed as

```
% mpirun -np 16 openmx SiC_Primitive.dat > sic_primitive.std &
```

The input file 'SiC_Primitive.dat' can be found in the directory 'work/unfolding_example', and the basis functions and geometrical structure are specified as

```
Species.Number       3
<Definition.of.Atomic.Species
C    C7.0-s2p2d1     C_PBE19
Si   Si7.0-s2p2d1    Si_PBE19
Te  Te11.0-s2p2d2f1  E
Definition.of.Atomic.Species>

Atoms.Number    3
Atoms.SpeciesAndCoordinates.Unit  FRAC  # Ang|AU
<Atoms.SpeciesAndCoordinates
1  C    0.33333333   0.66666666   0.50000000   2.0  2.0
2  Si   0.66666666   0.33333333   0.50000000   2.0  2.0
3  Te   0.00000000   0.00000000   0.50000000   0.0  0.0
Atoms.SpeciesAndCoordinates>

Atoms.UnitVectors.Unit             Ang   # Ang|AU
```

```
<Atoms.UnitVectors
3.0690  0.0000000000   0.000
-1.5345  2.6578319641   0.000
0.0000  0.0000000000  10.000
Atoms.UnitVectors>
```

where an empty atom having basis functions with a long tail is allocated at the center of the hexagon in order to improve description of conduction bands. Since the keyword 'Band.dispersion' is switched on as

```
Band.dispersion              on       # on|off, default=off
Band.Nkpath                   3
<Band.kpath
60 0.33333333333 0.33333333333 0.00000000000  0.00000000000 0.00000000000 0.00000000000 K G
52 0.00000000000 0.00000000000 0.00000000000  0.50000000000 0.00000000000 0.00000000000 G M
30 0.50000000000 0.00000000000 0.00000000000  0.33333333333 0.33333333333 0.00000000000 M K
Band.kpath>
```

you might be able to plot the band dispersion using 'sic_primitive.BANDDAT1' as the solid line shown in Fig. 61(a). As for plotting the band dispersion, please refer the section 'Band dispersion'.

**Keywords relevant to analysis of band structure**

For the calculation, the following keywords are also given as well

```
Unfolding.Electronic.Band      on       # on|off, default=off
Unfolding.LowerBound        -10.0       # default=-10 eV
Unfolding.UpperBound          6.0       # default= 10 eV

Unfolding.Nkpoint              4

<Unfolding.kpoint
K 0.33333333333 0.33333333333 0.0000000000
G 0.00000000000 0.00000000000 0.0000000000
M 0.50000000000 0.00000000000 0.0000000000
K 0.33333333333 0.33333333333 0.0000000000
Unfolding.kpoint>

Unfolding.desired_totalnkpt    30
```

The specification of the keywords above are listed below.

- **Unfolding.Electronic.Band** on| off, default=off

  To analyze and/or unfold bands the keyword should be switched on. The default is 'off'.

- **Unfolding.LowerBound**

  The keyword 'Unfolding.LowerBound' specifies the lower bound for the energy of bands in the analysis, where the energy is taken as the relative energy to the chemical potential. The default value is -10 eV.

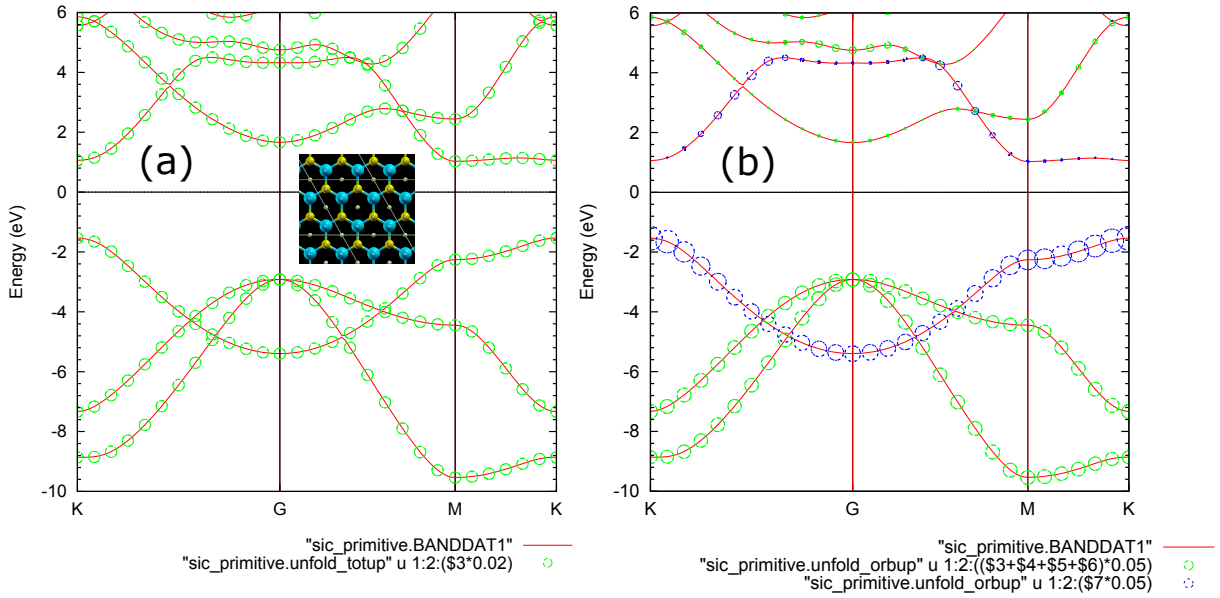



Figure 61: (a) Band structure of SiC *primitive* cell in a two-dimensional honeycomb structure without imperfection as shown in the inset, where the red line was obtained by the conventional calculation, and the green circles represent the total spectral weight. The radius reflects the magnitude of the weight. (b) Orbitally decomposed spectral weights of SiC *primitive* cell in a two-dimensional honeycomb structure without imperfection, where the green and purple circles represent the sum of spectral weights for the $s$, $p_x$, and $p_y$ orbitals, and the $p_z$ orbitals on the carbon atom, respectively. The radius reflects the magnitude of the weight. The calculation was performed by using an input file 'SiC_Primitive.dat' in the directory 'work/unfolding_example'.

- **Unfolding.UpperBound**

  The keyword 'Unfolding.UpperBound' specifies the upper bound for the energy of bands in the analysis, where the energy is taken as the relative energy to the chemical potential. The default value is 10 eV.

- **Unfolding.Nkpoint**

  The keyword 'Unfolding.Nkpoint' specifies the number of **k**-points appearing in the keyword 'Unfolding.kpoint'.

- **Unfolding.kpoint**

  The keyword 'Unfolding.kpoint' specifies the **k**-points of which number is given by the keyword 'Unfolding.Nkpoint'. In the above case four **k**-points are given, and three **k**-paths conecting the two **k**-points are considered, i.e., one from 'K' to 'G', one from 'G' to 'M', and one from 'M' to 'K'. Along the **k**-paths the analysis of bands is performed. Note that the unit for the **k**-points specified by the keyword is the reciprocal lattice vectors of the unit cell given by the keyword 'Atoms.UnitVectors' in this case, while the cell vectors can be changed to perform unfolding of bands by utilizing a keyword 'Unfolding.ReferenceVectors'. At this moment, it should be noted that the reciprocal lattice vectors of the unit cell given by the keyword 'Atoms.UnitVectors' is

automatically used unless the keyword 'Unfolding.ReferenceVectors' is explicitly specified. The keyword 'Unfolding.ReferenceVectors' will be explained in the next subsection.

- **Unfolding.desired_totalnkpt**

  The k-paths specified by the keyword 'Unfolding.kpoint' are divided with a nearly equal spacing, where the spacing is estimated by (the total length of all the k-paths)/Unfolding.desired_totalnkpt. Starting from the estimated spacing, the actual spacing is automatically adjusted from one k-path to another k-path so that the **k**-points specified by the keyword 'Unfolding.kpoint' can be always included in the analysis.

**Output files relevant to analysis of band structure**

After getting the SCF convergence, the following files related to analyzing and/or unfolding bands will be generated.

- sic_primitive.unfold_totup

  The total spectral weight given by Eq. (24) in Ref. [142] is stored. The first, second, and third columns correspond to the distance measured from the first $\underline{k}$-point given by the keyword 'Unfolding.kpoint' in the unit of Bohr$^{-1}$, energy relative to the chemical potential in the unit of eV, and the total spectral weight, respectively. In the spin-polarized calculation, 'System.Name.unfold_totdn' is also generaged for the down spin case, while only a single file 'System.Name.unfold_tot' is generated in the non-collinear calculation.

- sic_primitive.unfold_orbup

  The orbitally decomposed spectral weights given by Eq. (26) in Ref. [142] are stored. The first, second, and subsequent columns correspond to the distance measured from the first **k**-point given by the keyword 'Unfolding.kpoint' in the unit of Bohr$^{-1}$, energy relative to the chemical potential in the unit of eV, and the orbitally decomposed spectral weights, respectively. The sequence of the orbitally decomposed spectral weights can be found in 'System.Name.out'. In the spin-polarized calculation, 'System.Name.unfold_orbdn' is also generaged for the down spin case, while only a single file 'System.Name.unfold_orb' is generated in the non-collinear calculation.

- sic_primitive.unfold_plotexample

  As an example of plotting above the data by gnuplot, 'System.Name.unfold_plotexample' is generated. On the command line of the gnuplot, one can perform as

  ```
  gnuplot> load 'sic_primitive.unfold_plotexample'
  ```

  According to your purpose, it is needless to say that you can modify the file.

By plotting both 'sic_primitive.BANDDAT1' and 'sic_primitive.unfold_totup' as

```
gnuplot> set style data lines
gnuplot> set zeroaxis
gnuplot> set key below
gnuplot> set ytics 1
gnuplot> set mytics 5
```

```
gnuplot> set xra [0.000000:1.708883]
gnuplot> set yra [-10.0:6.0]
gnuplot> set ylabel "eV"
gnuplot> set xtics ("K" 0.000000, "G" 0.722258, "M" 1.347753, "K" 1.708882)
gnuplot> p "sic_primitive.BANDDAT1","sic_primitive.unfold_totup" u 1:2:($3*0.02) w circle
```

one may obtain a figure as shown in Fig. 61(a), where the solid line and circle correspond to "sic_primitive.BANDDAT1" and "sic_primitive.unfold_totup", respectively. The radius of the circle reflects the spectral weight, and all the radii are unity in this case, resulting in the equivalent radius for all the points, since we analyze the band dispersion represented by the Brillouin zone of the original cell.

Now let us move on 'sic_primitive.unfold_orbup' storing orbitally decomposed spectral weights. In a similar way above, one can plot the orbitally decomposed spectral weights as

```
gnuplot> p "sic_primitive.BANDDAT1","sic_primitive.unfold_orbup" u 1:2:(($3+$4+$5+$6)*0.05) w circle,
"sic_primitive.unfold_orbup" u 1:2:($7*0.05) w circle
```

Then, you may obtain a figure as shown in Fig. 61(b). In this case, the sum of spectral weights for the $s$, $p_x$, and $p_y$ orbitals, and the $p_z$ orbitals on the carbon atom are shown by the green and purple circles, respectively. It can be confirmed from the analysis that the $\sigma$ and $\pi$ bands are clearly distinguished. As for the format of 'sic_primitive.unfold_orbup', please refer the explanation above.

## 52.2 Unfolding of band structures

In the subsection, we show how the band structure calculated for a supercell can be unfolded into the Brillouin zone of a reference unit cell that a user specifies. As an example, let us consider again SiC in a two-dimensional honeycomb structure without imperfection. However, the unit cell in this case is extended to the $(2 \times 2)$ supercell as given by

```
Atoms.Number 12
Atoms.SpeciesAndCoordinates.Unit  FRAC  # Ang|AU
<Atoms.SpeciesAndCoordinates
 1 C    0.16666666   0.33333333   0.50000000  2 2
 2 C    0.66666666   0.33333333   0.50000000  2 2
 3 C    0.16666666   0.83333333   0.50000000  2 2
 4 C    0.66666666   0.83333333   0.50000000  2 2
 5 Si   0.33333333   0.16666666   0.50000000  2 2
 6 Si   0.83333333   0.16666666   0.50000000  2 2
 7 Si   0.33333333   0.66666666   0.50000000  2 2
 8 Si   0.83333333   0.66666666   0.50000000  2 2
 9 Te   0.00000000   0.00000000   0.50000000  0 0
10 Te   0.50000000   0.00000000   0.50000000  0 0
11 Te   0.00000000   0.50000000   0.50000000  0 0
12 Te   0.50000000   0.50000000   0.50000000  0 0
Atoms.SpeciesAndCoordinates>

Atoms.UnitVectors.Unit             Ang   # Ang|AU
<Atoms.UnitVectors
```

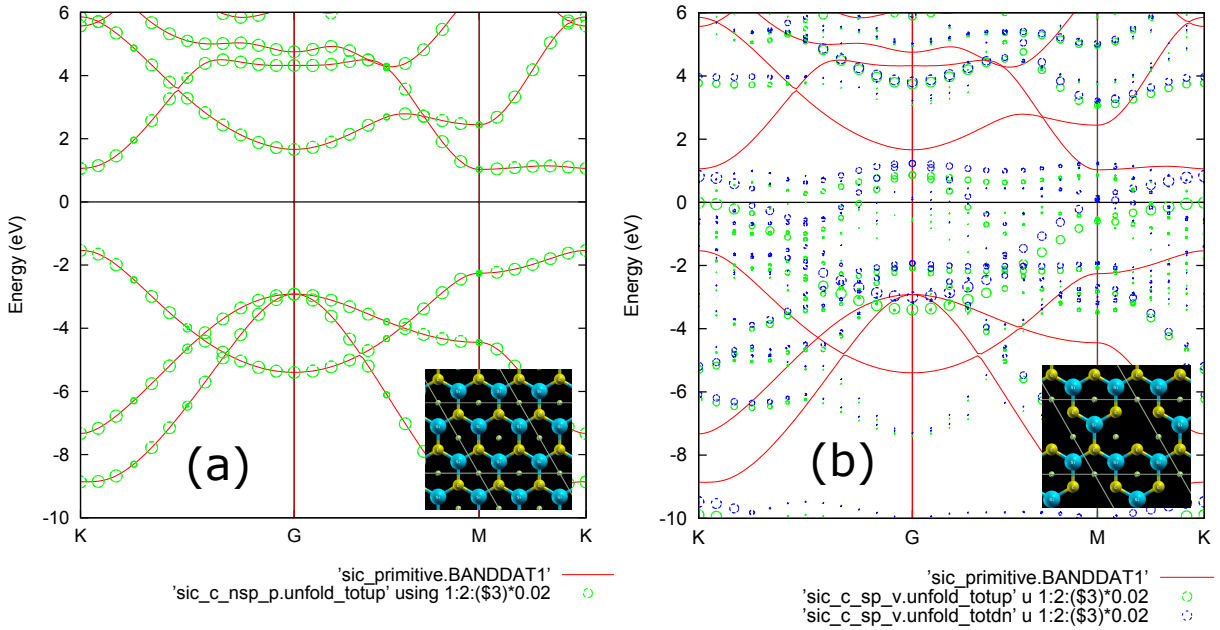



Figure 62: (a) Band structure of a SiC ($2 \times 2$) *supercell* in a two-dimensional honeycomb structure without imperfection as shown in the inset, where the red line was obtained by the conventional calculation for the *primitive* cell, and the green circles represent the total spectral weight obtained by unfolding. The radius reflects the magnitude of the weight. The calculation was performed by using an input file 'SiC_C_NSP_P.dat' in the directory 'work/unfolding_example'. (b) Band structure of a SiC ($2 \times 2$) *supercell* in a two-dimensional honeycomb structure with a Si vacancy as shown in the inset, where the red line was obtained by the conventional calculation for the *primitive* cell, and the green and blue circles represent the total spectral weight obtained by the unfolding procedure for up- and down-spin states, respectively. The radius reflects the magnitude of the weight. The calculation was performed by using an input file 'SiC_C_SP_V.dat' in the directory 'work/unfolding_example'.

```
  6.138   0.0000000000   0.00
 -3.069   5.3156639282   0.00
  0.000   0.0000000000  10.00
Atoms.UnitVectors>
```

The SCF calculation for the supercell of the two-dimensional SiC can be performed as

```
% mpirun -np 16 openmx SiC_C_NSP_P.dat > sic_c_nsp_p.std &
```

where the input file 'SiC_C_NSP_P.dat' can be found in the directory 'work/unfolding_example'. After finishing the SCF calculation, you obtain the following files relevant to the unfolding calculation:

```
sic_c_nsp_p.unfold_totup
sic_c_nsp_p.unfold_orbup
sic_c_nsp_p.unfold_plotexample
```

By plotting 'sic_c_nsp_p.unfold_totup' with gnuplot together with the band structure of the primitive cell as a reference, you may obtain a figure as shown in Fig. 62(a). It is confirmed that the unfolded bands of the supercell exactly recover those of the primitive cell as expected.

To perform the unfolding of bands in the calculation, the following keywords are specified in addition to the keywords explained in the previous subsection:

```
<Unfolding.ReferenceVectors
 3.0690  0.0000000000   0.000
-1.5345  2.6578319641   0.000
 0.0000  0.0000000000  10.000
Unfolding.ReferenceVectors>

<Unfolding.Map
 1 1
 2 1
 3 1
 4 1
 5 2
 6 2
 7 2
 8 2
 9 3
10 3
11 3
12 3
Unfolding.Map>
```

The specification of the keywords above are explained below.

- **Unfolding.ReferenceVectors**

  With the keyword 'Unfolding.ReferenceVectors' one can define a reference unit cell for which the unfolding of bands is performed based on Eq. (24) in Ref. [142]. In the example above, the primitive cell was used as the reference cell. The format is the same as that for the keywords 'Atoms.UnitVectors', i.e., the first, second, and third lines correspond to **a**-, **b**-, and **c**-axes, respectively. It is also noted that the unit of the reference unit cell is the same as that used for the supercell, which is specified by the keyword 'Atoms.UnitVectors.Unit'. In the subsection 'Analysis of band structures', the keyword 'Unfolding.ReferenceVectors' was not specified explicitly. In such a case, 'Atoms.UnitVectors' is automatically set for 'Unfolding.ReferenceVectors'.

- **Unfolding.Map**

  The keyword 'Unfolding.Map' specifies how atoms in the supercell can be mapped to atoms in the reference cell. The first and second columns are the serial number of atom, which corresponds to the number of the first column in the specification of the keyword 'Atoms.SpeciesAndCoordinates', and an identification number representing the group to which the atom belongs. In the example above, atoms 1 to 4 belong to the group 1, and atoms 5 to 8 the group 2, and atoms 9 to 12 the group 3. Based on the information given by the keyword 'Unfolding.Map', the relabelling of the index for each atom is performed as given by Eq. (17) in Ref. [142], which is the essential part in the unfolding procedure. It is also noted that the identification number should be positive

and integer, while the the integer needs not to start from '1', and not to appear in ascending order.

Here we show one more example, i.e., a SiC ($2 \times 2$) *supercell* in a two-dimensional honeycomb structure with a Si vacancy. The SCF calculation can be performed by

```
% mpirun -np 16 openmx SiC_C_SP_V.dat > sic_c_sp_v.std &
```

where the input file 'SiC_C_SP_V.dat' can be found in the directory 'work/unfolding_example'. By removing a Si atom from the ($2 \times 2$) *supercell* cell structure, the honeycomb structure with a Si vacancy was created as

```
Atoms.Number 11
Atoms.SpeciesAndCoordinates.Unit  FRAC  # Ang|AU
<Atoms.SpeciesAndCoordinates
1 C    0.16666666   0.33333333   0.50000000  2.5 1.5
2 C    0.66666666   0.33333333   0.50000000  2.5 1.5
3 C    0.16666666   0.83333333   0.50000000  2.5 1.5
4 C    0.66666666   0.83333333   0.50000000  2.5 1.5
5 Si   0.33333333   0.16666666   0.50000000  2.5 1.5
6 Si   0.83333333   0.16666666   0.50000000  2.5 1.5
7 Si   0.33333333   0.66666666   0.50000000  2.5 1.5
8 Te   0.00000000   0.00000000   0.50000000  0.0 0.0
9 Te   0.50000000   0.00000000   0.50000000  0.0 0.0
10 Te   0.00000000   0.50000000   0.50000000  0.0 0.0
11 Te   0.50000000   0.50000000   0.50000000  0.0 0.0
Atoms.SpeciesAndCoordinates>
```

The mapping of atoms in the supercell to those in the reference cell was specified by

```
<Unfolding.Map
1 1
2 1
3 1
4 1
5 2
6 2
7 2
8 3
9 3
10 3
11 3
Unfolding.Map>
```

After finishing the SCF calculation, you obtain the following files relevant to the unfolding calculation:

```
sic_c_sp_v.unfold_totup
sic_c_sp_v.unfold_totdn
```

```
sic_c_sp_v.unfold_orbup
sic_c_sp_v.unfold_orbdn
sic_c_sp_v.unfold_plotexample
```

By plotting 'sic_c_sp_v.unfold_totup' and 'sic_c_sp_v.unfold_totdn' with gnuplot together with the band structure of the primitive cell as a reference, you may obtain a figure as shown in Fig. 62(b). It is found from the unfolded spectral weights that the characteristic feature of the perfect case is still preserved in spite of introduction of the vacancy, and the chemical potential is pushed up largely. We also see that the electronic state is spin-polarized due to dangling bonds of carbon atoms. The analysis shows that the unfolding method would be useful to analyze how the original bands are pertubed by the introduced imperfection.

When you consider an impurity instead of introduction of vacancy, you only have to assign the impurity with an identification number. For example, if you introduce an impurity of atom 13 in the SiC $(2 \times 2)$ supercell, you can define the mapping rule in relabelling as

```
<Unfolding.Map
  1 1
  2 1
  3 1
  4 1
  5 2
  6 2
  7 2
  8 2
  9 3
 10 3
 11 3
 12 3
 13 4
Unfolding.Map>
```

In case of multi-impurities and existence of surfaces, you can define the mapping in a similar way above.

### 52.3 The origin of the reference unit cell

If you take a close look at Eq. (17) in Ref. [142], you may notice that the relabelling requires the two mapping rules, i.e., $R \to R + r_0(M)$ and $M \to m'(M)$, where $R$ and $r_0(M)$ are the lattice vectors of the supercell and the reference cells, respectively, and $M$ and $m'(M)$ are a symbolic atomic orbital index, respectively. We have already given the keyword 'Unfolding.Map' which actually gives the mapping rule in relabelling $M \to m'(M)$. In this subsection, we explain how the relabelling $R \to R + r_0(M)$ is taken into account. The relabelling $R \to R + r_0(M)$ depends on how the reference unit cell is placed relative to the atomic coordinates in real space, which is equivalent to the choice of the origin of the reference unit cell. The choice of the origin can be insensitive to the unfolded weights, because the mapping rule in relabelling $M \to m'(M)$ does not change in most cases. However, the mapping rule may vary depending on the choice of the origin in subtle cases such as surfaces and

largely distorted structures. As default, OpenMX will try to estimate an origin using the following two rules. The first rule is that no more than one atom labeled by the same identification number in the keyword 'Unfolding.Map' can be assigned in each reference cell. The multiple assignment of atoms labeled by the same identification number to a reference cell may happen in a largely distorted structure. OpenMX will automatically try to avoid such a situation. The second rule is that the origin of the reference cell is determined by minimizing the total number of the reference lattices that the number of atoms allocated by the rellabelling is non-zero. Since a system with surfaces represented by the supercell approach has vacuum region between the slabs, the total number of the reference lattices having non-zero allocated atoms may vary depending on the choice of origin. OpenMX will automatically try to minimize the total number of the reference lattices having non-zero allocated atoms. Applying the two rules may satisfy requirement for most of users, however, you may want to control the origin by yourself. For this purpose, the following keyword is available:

```
<Unfolding.ReferenceOrigin
  0.1 0.2 0.3
Unfolding.ReferenceOrigin>
```

where the unit is defined by the keyword 'Atoms.UnitVectors.Unit'.

## 52.4 Intensity map of unfolded spectral weight

The unfolded spectral weight can be visualized by an intensity map that the weight $w$ is smeared out by a Lorentian function:

$$L(k,E) = \frac{w}{(k/\Delta_k)^2 + (E/\Delta_E)^2 + 1},$$

where $k$ and $E$ are the magnitude of **k**-vector in Bohr$^{-1}$ and energy in eV, respectively, and $\Delta_k$ and $\Delta_E$ are the corresponding degree of smearing. Note that the absolute value of the intensity map does not have physical meaning. The visualization can be performed by the following three steps:

### (1) Compilation of intensity_map.c

In the the directory 'source', please compile 'intensity_map.c' as

```
gcc intensity_map.c -lm -o intensity_map
```

and copy the executable file 'intensity_map' to your work directory.

### (2) Generation of the intensity map

After finising the unfolding calculation, you can generate a file storing a mesh data for drawing the intensity map using 'intensity_map'. For the case of SiC ($2 \times 2$) *supercell* in a two-dimensional honeycomb structure with a Si vacancy discussed in the previous subsection, where the input file is 'SiC_C_SP_V.dat', e.g., one can generate a file 'sic-intmap.txt ' storing the mesh data by

```
./intensity_map sic_c_sp_v.unfold_totup -c 3 -k 0.1 -e 0.1 -l -10 -u 6 > sic-intmap.txt
```

where the arguments have the following meaning:

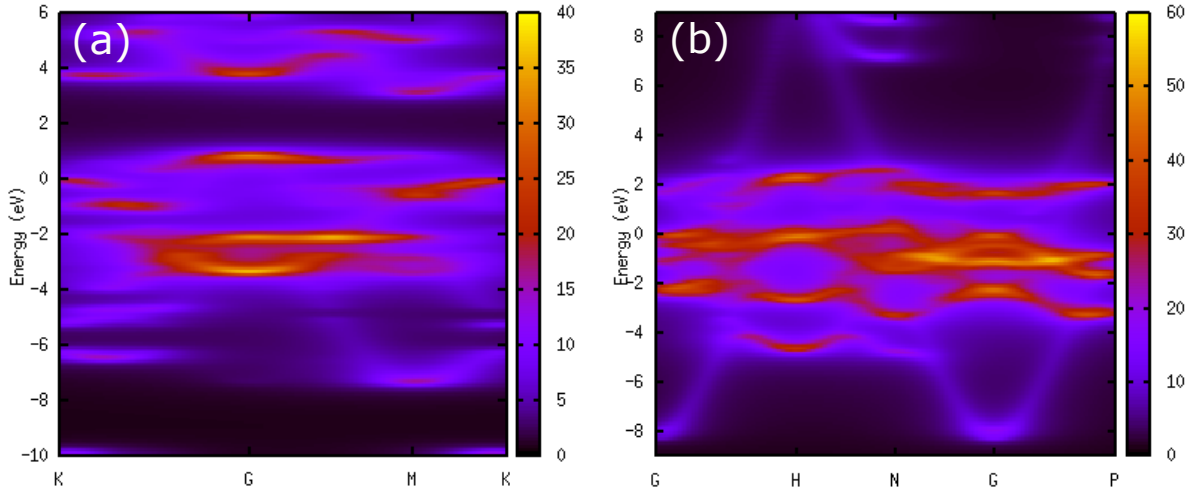



Figure 63: (a) Intensity map of the unfolded total spectral weight for the up-spin state of a SiC $(2\times2)$ supercell in a two-dimensional honeycomb structure with a Si vacancy, where the SCF calculation was performed by 'SiC_C_SP_V.dat' with a modification that the keyword 'Unfolding.desired_totalnkpt' is set to 300 for better resolution. The degree of smearings are 0.1 (Bohr$^{-1}$) and 0.1 (eV) for $\Delta_k$ and $\Delta_E$, respectively. (b) Intensity map of the unfolded total spectral weight of the cubic cell of Fe in the BCC structure with structural disorder, where the SCF calculation was performed by BCC_Fe_N_SO_Disorder.dat with a modification that the keyword 'Unfolding.desired_totalnkpt' is set to 300 for better resolution. The degree of smearings are 0.1 (Bohr$^{-1}$) and 0.1 (eV) for $\Delta_k$ and $\Delta_E$, respectively.

```
-c     column of spectral weight you analyze
-k     degree of smearing (Bohr^{-1}) in {\bf k}-vector
-e     degree of smearing (eV) in energy
-l     lower bound of energy for drawing the map
-u     upper bound of energy for drawing the map
```

You might be confused by the argument '-c' specifying the column number in the file. When you analyze 'System.Name.unfold_orbup(dn)', you will refer the sequence number for pseudo-atomic orbitals in 'System.Name.out'. Then, it should be noted that the number $N_{\rm col}$ specified by '-c' is related to the sequence number $N_{\rm seq}$ for pseudo-atomic orbitals in 'System.Name.out' by $N_{\rm col} = N_{\rm seq} + 2$.

### (3) Drawing of the intensity map

Using gnuplot you can draw the intensity map. For example, for the calculation with the input file 'SiC_C_SP_V.dat' it can be done as follows:

```
gnuplot> set yrange [-10.000000:6.000000]
gnuplot> set ylabel 'Energy (eV)'
gnuplot> set xtics('K' 0.000000,'G' 0.722259,'M' 1.347753,'K' 1.708883)
gnuplot> set xrange [0:1.708883]
gnuplot> set arrow nohead from 0,0 to 1.708883,0
```

```
gnuplot> set arrow nohead from 0.722259,-10.000000 to 0.722259,6.000000
gnuplot> set arrow nohead from 1.347753,-10.000000 to 1.347753,6.000000
gnuplot> set pm3d map
gnuplot> sp 'sic-intmap.txt'
```

Then, you may obtain a figure as shown in Fig. 63(a).

### 52.5 In case of non-collinear DFT calculations

The functionality for unfolding bands is also supported for the non-collinear DFT calculations with spin-orbit coupling, and compatible with any other functionalities such as the plus U method. For such cases, there is no additional keyword.

### 52.6 Examples

For user's convenience, input files for six examples can be found in 'work/unfolding_example' as follows:

- SiC_Perfect.dat

  The primitive cell of SiC in a two-dimensional honeycomb structure without imperfection for analyzing how each band consists of pseudo-atomic orbitals, where the collinear non-spin polarized calculation is performed.

- SiC_C_NSP_P.dat

  The $(2 \times 2)$ supercell of SiC in a two-dimensional honeycomb structure without imperfection for unfolding bands, where the collinear non-spin polarized calculation is performed.

- SiC_C_SP_V.dat

  The $(2 \times 2)$ supercell of SiC in a two-dimensional honeycomb structure with a Si vacancy for unfolding bands, where the collinear spin polarized calculation is performed.

- SiC_NC_SO_P.dat

  The $(2 \times 2)$ supercell of SiC in a two-dimensional honeycomb structure without imperfection for unfolding bands, where the non-collinear calculation is performed with spin-orbit coupling.

- SiC_NC_SO_V.dat

  The $(2 \times 2)$ supercell of SiC in a two-dimensional honeycomb structure with a Si vacancy for unfolding bands, where the non-collinear calculation is performed with spin-orbit coupling.

- BCC_Fe_Perfect.dat

  The primitive cell of Fe in the BCC structure without imperfection for analyzing how each band consists of pseudo-atomic orbitals, where the collinear spin polarized calculation is performed.

- BCC_Fe_C_SP_Perfect.dat

  The cubic cell of Fe in the BCC structure without imperfection for unfolding bands, where the collinear spin polarized calculation is performed.

- BCC_Fe_N_SO_Disorder.dat

  The cubic cell of Fe in the BCC structure with structural disorder for unfolding bands, where the non-collinear calculation is performed with spin-orbit coupling.

# 53 Analysis of spin texture in the k-space

Spin splitting in the band structure as can be seen in the Rashba effect may occur when the spin-orbit coupling is taken into account. The spin splitting can be resolved in each state and wave vector **k**, and the state- and **k**-resolved spin splitting, which is a spin structure on the band dispersion relation in the reciprocal space, is called spin texture. Using a post-processing code 'kSpin' one can calculate the spin texture in the case of a non-collinear calculation with spin-orbit coupling. The state- and **k**-resolved spin density matrix, which is referred to as the k-space spin density matrix hereafter, is calculated from the two component spinor, and takes a form of a $2 \times 2$ matrix. The spin texture is actually calculated using the $2 \times 2$ matrix. In addition to the spin texture, 'kSpin' provides us the k-space spin density matrix so that physical origins of phenomena caused by spin-orbit interaction such as the Rashba effect can be analyzed. 'kSpin' also supports the analysis of spin texture resolved for not only each state and wave vector **k**, but also atom and pseudo-atomic orbital, which may help us to understand which atom and orbital play a central role of the spin splitting. Note that 'kSpin' is applicable to not only the Rashba effect but also topological insulators and systems with non-Rashba type spin splitting.

In the following subsequent subsections, we explain how 'kSpin' can be used for such an analysis with instructive examples of calculations. It should be noted that 'kSpin' is compatible with only the non-collinear case, and not supported for the collinear case. See also the section of 34 'Non-collinear DFT' to get information of non-collinear calculations.

To acknowledge in any publications by using the functionality, the citation of the reference [78] would be appreciated. Also, a technical note regarding the implementation of the functionality is available at http://www.openmx-square.org/tech_notes/note_kSpin-1_0.pdf.

## 53.1 General

In this subsection, the calculation of spin textures is illustrated with a simple model of an Au(111) surface. The spin texture/k-space spin density matrix are analyzed by the following two or three steps:

### 1. SCF calculation

First, please perform a conventional SCF calculation using an input file 'Au111Surface_FL.dat' stored in the directory 'work'. Then, the following keywords 'scf.SpinPolarization' and 'HS.fileout' should be switched to 'NC' and 'ON', respectively, as follows:

```
scf.SpinPolarization         NC    # On|Off|NC
HS.fileout                   ON    # on|off, default=Off
```

Also, when you consider to investigate spin texture, the keyword 'scf.SpinOrbit.Coupling' should be switched on:

```
scf.SpinOrbit.Coupling       ON    # On|Off
```

Once the calculation is completed normally, you can obtain an output file 'Au111Surface.scfout' in the directory 'work'. Figure 64 shows the band structure of the Au(111) surface.

### 2. Calculation of the spin texture and k-space spin density matrix

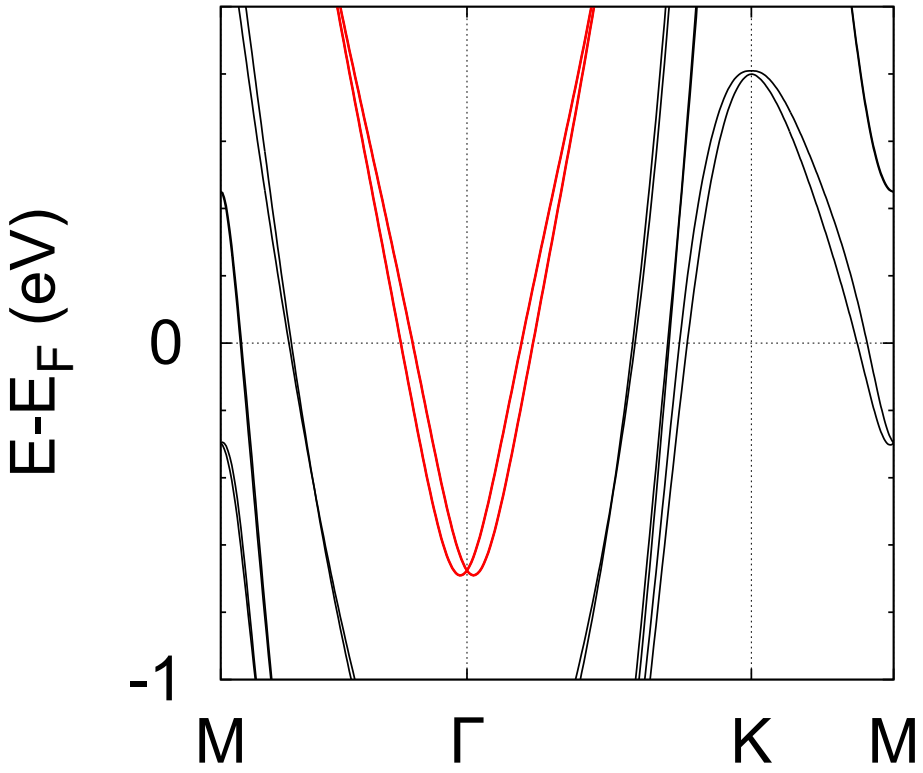



Figure 64: Band structure for the Rashba spin splitting at the Au(111) surface. This figure is obtained by a Band file 'Au111Surface.Band' (See the section of 19 Band dispersion). The red curves show the Rashba bands, which correspond to band indices 55 and 56. This highlight of the Rashba bands can be reproduced by modifing a GNUBAND file 'Au111Surface.GNUBAND' or using OMXTool [146].

Let us analyze Rashba bands with a spin splitting shown in Fig. 64. The spin texture and k-space spin density matrix are calculated by a post-processing code 'kSpin'. The executable file can be obtained by compilation in the directory 'source' as follows:

```
% make kSpin
```

After the successful compilation, you can find the executable file 'kSpin' in the directory 'work'. Then, please move to the directory 'work', and perform as follows:

```
% ./kSpin Au111Surface_FL.dat
```

Note that the input file must include appropriate keywords about 'kSpin', which will be explained later on. 'kSpin' has four ways to calculate the k-space spin density matrix: *FermiLoop*, *GridCalc*, *BandDispersion*, and *MulPOnly*. The details of the four ways are provided in each subsection. The example 'Au111Surface_FL.dat' is for *FermiLoop* so that you can proceed with this exercise by moving to the subsection of 53.2 *FermiLoop*.

**3. (Optional) Analysis of the k-space spin density matrix**

You may need to analyze the k-space spin density matrix, or atomic decomposition of it. Moreover, it can be decomposed into the *s*-, *p*-, *d*-, or *f*-character components. You can use a post-processing code 'MulPCalc' to analyze the k-space spin density matrix resolved atom and pseudo-atomic orbital. How to do that will be explained in the subsection of 53.6 *MulPCalc*.

## 53.2 FermiLoop: Calculation on a constant-energy level

'kSpin' has four ways to calculate the k-space spin density matrix, which can be specified by a keyword 'Calc.Type'. Here we introduce *FermiLoop* among the four ways. *FermiLoop* can calculate the spin texture/k-space spin density matrix on a constant-energy level (e.g. the Fermi level). The calculation of spin texture using *FermiLoop* is illustrated here with a simple model of an Au(111) surface. *FermiLoop* searches k-points by two steps: In the first step, *FermiLoop* performs a rough search to find the bands to calculate; In the second step, *FermiLoop* detects k-points on the constant-energy level using a triangular mesh.

After the calculation of 'Au111Surface_FL.dat' as explained above, you may try to run 'kSpin'. The keywords relevant to the executation of kSpin can be found at the bottom of the input file 'Au111Surface_FL.dat' stored in the directory 'work' as shown below:

### List of keywords relevant to kSpin

```
Filename.scfout      Au111Surface.scfout
Filename.outdata     Au111Surface_FL
Calc.Type            FermiLoop         # FermiLoop, GridCalc,
                                         BandDispersion, or MulPOnly
                                         default: MulPOnly
Energy.Range         0.0  0.0          # eV; default: 0.0  0.0
Search.kCentral      0.0  0.0  0.0     # default: 0.0  0.0  0.0
Calc.Type.3mesh      2                 # default: 1
kRange.3mesh         0.2  0.2          # default: 0.5  0.5
k-plane.1stStep      21 21             # default: 2 2
k-plane.2ndStep      3 3               # default: 3 3
Eigen.Brent          On                # on|off, default: On
Trial.Brent          5                 # default: 5
Calc.Bandbyband      Off               # on|off, default: Off
Calc.Band.Min        55
Calc.Band.Max        56
MulP.Vec.Scale       0.1  0.1  0.1     # default: 1.0  1.0  1.0
```

### Specification of keywords

The specification of each keyword is explained below:

**Filename.scfout**
Specify the name of the scfout file which will be read by 'kSpin'.

**Filename.outdata**

Specify a name for output files. This keyword corresponds to the keyword 'System.Name' for OpenMX calculations.

**Calc.Type**
Choose either *FermiLoop*, *GridCalc*, *BandDispersion*, or *MulPOnly*. The default setting is *MulPOnly*. Here we choose *FermiLoop* for the exercise.

**Energy.Range**
The keyword specifies the energy range in which bands to be analyzed are searched. For *FermiLoop*, specify the two same values for an energy level. The unit is 'eV'. If different values are set, the average of them will be used. The default is '-0.5 0.5' (i.e. the Fermi level).

**Search.kCentral**
Specify a set of three values for the central k-point around which *FermiLoop* should search k-points on the specified constant-energy level (i.e. values for the keyword 'Energy.Range'). The notation to specify the k-point follows the keyword 'Band.kpath'. The default is '0.0 0.0 0.0' (i.e. Γ-point).

**Calc.Type.3mesh**
Specify a plane on which the spin texture is calculated. Set a value '1', '2', and '3', for the case of $k_a k_b$-, $k_b k_c$-, and $k_c k_a$-planes, respectively. The default is '1'.

**kRange.3mesh**
Specify two values for a two-dimensional domain in the reciprocal space where k-points should be calculated. For example, if the value for the keyword 'Calc.Type.3mesh' is '1' ($k_a k_b$-plane), values '0.2 0.3' specifies a domain: $-0.2 \leq k_a \leq 0.2$, $-0.3 \leq k_b \leq 0.3$. The notation of the k-points follows the keyword 'Band.kpath'. The default is '0.5 0.5' (i.e. the whole of the first brillouin zone).

**k-plane.1stStep**
Specify the number of grid points to divide the domain by the keyword 'kRange.3mesh' for the first search. For example, if the value for the keyword 'Calc.Type.3mesh' is '1' ($k_a k_b$-plane), values '2 3' specifies the number of grid points as follows: 2 for $k_a$-axis, 3 for $k_b$-axis. If the values are '1 1' and the value for the keyword 'Calc.Bandbyband' is 'ON', *FermiLoop* will omit the first step, which is useful for a large-scale calculation with many MPI processes (cf. k-plane.2ndStep; Calc.Bandbyband; Calc.BandMin; Calc.BandMax). The default is '2 2'.

**k-plane.2ndStep**
Specify the number of grid points for the second search, which divides the subspace divided in the first search. The notation is the same as that for the keyword 'k-plane.1stStep' (cf. k-plane.1stStep). The default is '3 3'.

**Eigen.Brent**
Specify if *FermiLoop* should use the Brent method (ON) or not (OFF) to search k-points on the specified constant-energy level (i.e. values for the keyword 'Energy.Range'). If the value is 'OFF', *FermiLoop* will use a linear interpolation scheme. The default is 'ON' (cf. Trial.Brent).

**Trial.Brent**

Specify the maximum number of steps for the Brent method (cf. Eigen.Brent). This keyword is valid when the value for the keyword 'Eigen.Brent' is 'ON'. The default is '5' (cf. Eigen.Brent).

**Calc.Bandbyband**
Specify if *FermiLoop* should calculate given bands (ON) or not (OFF). (cf. Calc.BandMin; Calc.BandMax). The default is 'OFF'.

**Calc.BandMin**
Set the lower limit on the range for bands to calculate by specifying the band index. This keyword is valid when the value for the keyword 'Calc.Bandbyband' is 'ON' (cf. Calc.Bandbyband; Calc.BandMax). You can see band indices using OMXTool [146] or 53.4 *BandDispersion*.

**Calc.BandMax**
Set the upper limit on the range for bands to calculate by specifying the band index. This keyword is valid when the value for the keyword 'Calc.Bandbyband' is 'ON' (cf. Calc.Bandbyband; Calc.BandMin). You can see band indices using [146] or 53.4 *BandDispersion*.

**MulP.Vec.Scale**
Specify a scale to draw vectors expressing the spin texture. For example, values '0.1 0.2 0.3' specifies the scale as follows: 0.1 for x-axis, 0.2 for y-axis, 0.3 for z-axis. This keyword affects only 'XXXXX.Pxyz_YY' (XXXXX = the value for the keyword 'Filename.outdata'; YY = the band index). The default is '1.0 1.0 1.0'.

## Calculation

The spin texture/k-space spin density matrix are calculated by a post-processing code 'kSpin' in the directory 'work'. Then, please move to the directory 'work', and perform a calculation as follows:

```
% ./kSpin Au111Surface_FL.dat
```

or for the MPI calculation, for example, the case with 4 MPI processes

```
% mpirun -np 4 ./kSpin Au111Surface_FL.dat
```

As the calculation proceeds, you may see the following standard output:

```
*****************************************************************
*****************************************************************
 kSpin:
 code for evaluating spin related properties
 in momentum space of solid state materials.
 Copyright (C), 2019,
 Hiroki Kotaka, Naoya Yamaguchi and Fumiyuki Ishii.
 This software includes the work that is distributed
 in version 3 of the GPL (GPLv3).

 Please cite the following article:
 H. Kotaka, F. Ishii, and M. Saito,
```

```
 Jpn. J. Appl. Phys. 52, 035204 (2013).
 DOI: 10.7567/JJAP.52.035204.
*******************************************************************
*******************************************************************


Input filename is "Au111Surface.scfout"


Start "FermiLoop" Calculation (5).

########### ORBITAL DATA ##################
ClaOrb_MAX[0]:   2
ClaOrb_MAX[1]:   8
Total Band (2*n): 124
Central (  0.000000   0.000000   0.000000)
##########################################

########### EIGEN VALUE ###################
The number of BANDs    2 (  55->  56)
########### CONTOUR CALC ##################
  k-height :   0   0.000000
The number of BANDs    2 (  55->  56)
 l=  55,   k_points: 139 (array: 277)
 l=  56,   k_points: 115 (array: 229)
Total MulP data: 254
##########################################

##########################################

############ CALC TIME ####################
  Total Calculation Time: 26.851349 (s)
        Eigen Value Calc:  4.277838 (s)
 l=  55:    Contour Calc: 11.349228 (s)
               MulP Calc:  1.120379 (s)
 l=  56:    Contour Calc:  9.179602 (s)
               MulP Calc:  0.920524 (s)
##########################################
############ CALC TIME ####################
  Total Calculation Time: 26.869150 (s)
##########################################
```

When the calculation is completed normally as shown above, you can find the following output files in the directory 'work':

```
Au111Surface_FL.FermiSurf_53
```

```
Au111Surface_FL.Pxyz_53
Au111Surface_FL.FermiSurf_54
Au111Surface_FL.Pxyz_54
Au111Surface_FL.FermiSurf_55
Au111Surface_FL.Pxyz_55
Au111Surface_FL.FermiSurf_56
Au111Surface_FL.Pxyz_56
Au111Surface_FL.AtomMulP
Au111Surface_FL.MulP_s
Au111Surface_FL.MulP_p
Au111Surface_FL.MulP_p1
Au111Surface_FL.MulP_p2
Au111Surface_FL.MulP_p3
Au111Surface_FL.MulP_d
Au111Surface_FL.MulP_d1
Au111Surface_FL.MulP_d2
Au111Surface_FL.MulP_d3
Au111Surface_FL.MulP_d4
Au111Surface_FL.MulP_d5
Au111Surface_FL.plotexample
Au111Surface_FL.atominfo
temporal_12345.input
```

As an example, by executing the following command, you can obtain a figure of spin texture for the Rashba spin splitting in the Au(111) surface as shown in Fig. 65. which exhibits a typical Rashba-type spin texture.

```
% gnuplot Au111Surface_FL.plotexample
```

## Output files

The content of each output file is explained below:

**FermiSurf_YY file**
This file stores data of the k-points for the band with the band index YY searched on the specified constant-energy level. The first, second, and third columns correspond to the $k_x$, $k_y$, and $k_z$ components of the k-points in units of Bohr$^{-1}$, respectively. The fourth, and fifth columns correspond to the band index, the energy (in units of eV), respectively.

**Pxyz_YY file**
This file stores data of the expectation value of the Pauli matricies vectors for each k-point stored in the FermiSurf_YY file. The first, second, and third columns correspond to the $k_x$, $k_y$, and $k_z$ components of the k-points in units of Bohr$^{-1}$, respectively. The fourth, fifth, and sixth columns correspond to the expectation value of the $\sigma_x$, $\sigma_y$, and $\sigma_z$ in units of the Bohr magneton, respectively.

**AtomMulP file**

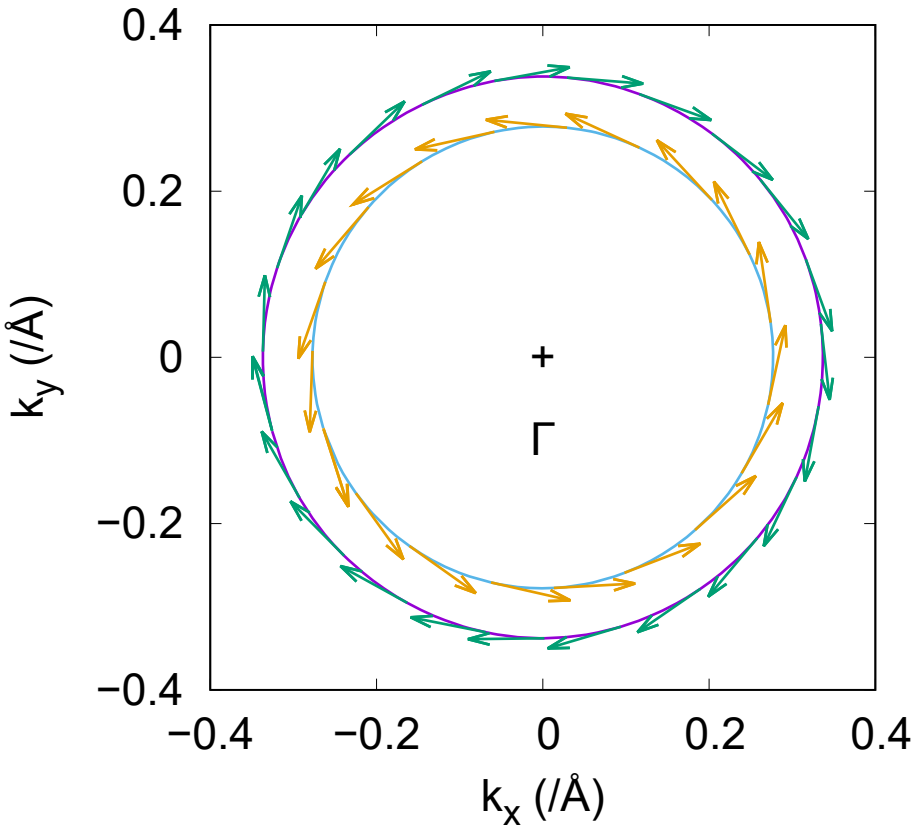



Figure 65: Spin textures for the Rashba spin splitting around Γ-point at the Au(111) surface. The outer and inner ones correspond to the band indices 55 and 56. The vectors represent the expectation values of the Pauli matrices vectors at each k-point stored in Pxyz_YY files 'Au111Surface_FL.Pxyz_55' and 'Au111Surface_FL.Pxyz_56'. The closed curves represent the constant-energy level stored in FermiSurf_YY files 'Au111Surface_FL.FermiSurf_55' and 'Au111Surface_FL.FermiSurf_56'. The central cross point corresponds to Γ-point. You can get this figure by a plotexample file 'Au111Surface_FL.plotexample'.

This file stores data of the k-space spin density matrix resolved to the atomic contribution, which can be analyzed by 53.6 *MulPCalc*.

**MulP_xx file**
This file stores data of the xx-component of the k-space spin density matrix resolved to the atomic contribution, and can be analyzed by 53.6 *MulPCalc*.

**plotexample file**
This file supplies an example of gnuplot scripts.

**atominfo file**
This file supplies information of lattice vectors and PAOs.

**temporal_12345.input**
This file is a copy of the input file stored in the scfout file.

### (Optional) Analysis of the k-space spin density matrix

*MulPCalc* can extract data to analyze the k-space spin density matrix from AtomMulP files or MulP_xx files. You can find an example of input files for *MulPCalc* at the bottom of an input file 'Au111Surface_FL.dat' and proceed to the analysis of the k-space spin density matrix after the above calculation as follows:

```
% ./MulPCalc Au111Surface_FL.dat
```

For more information of *MulPCalc*, see also the subsection of 53.6 *MulPCalc*.

### Other tips

*GridCalc* may be useful in finding appropriate settings for FermiLoop because *GridCalc* can draw spin the texture on a k-point grid to investigate a wider domain in the reciprocal space. Setting 'OFF' for the keyword 'Eigen.Brent' may be effective in reducing computational time. In this case, you should use a fine mesh by increaing the values for the keywords 'k-plane.1stStep' or 'k-plane.2ndStep' for accuracy.

## 53.3 GridCalc: Calculation on a k-point grid

'kSpin' has four ways to calculate the k-space spin density matrix, which can be specified by a keyword 'Calc.Type'. Here we introduce *GridCalc* among the four ways. *GridCalc* can calculate the spin textures/k-space spin density matrix on a given k-point grid for each band within a user-specified energy range. You can also specify bands to calculate instead of specifying an energy range. The calculation of spin textures using *GridCalc* is illustrated here with a simple model of an Au(111) surface.

After a calculation of OpenMX with an input file 'Au111Surface_GC.dat' stored in the directory 'work', you may try to run 'kSpin'. The keywords relevant to the executation of kSpin can be found at the bottom of the input file 'Au111Surface_GC.dat' stored in the directory 'work' as shown below:

### List of keywords relevant to kSpin

```
Filename.scfout      Au111Surface.scfout
Filename.outdata     Au111Surface_GL
Calc.Type            GridCalc          # FermiLoop, GridCalc,
                                         BandDispersion, or MulPOnly
                                         default: MulPOnly
Energy.Range        -1.0  1.0          # eV; default: 0.0  0.0
Search.kCentral      0.0  0.0  0.0     # default: 0.0  0.0  0.0
Calc.Type.3mesh      2                 # default: 1
kRange.3mesh         0.5  0.5          # default: 0.5  0.5
k-plane.1stStep      14 14             # default: 2 2
Calc.Bandbyband      Off               # default: Off
Calc.Band.Min        55
Calc.Band.Max        56
MulP.Vec.Scale       0.1  0.1  0.1     # default: 1.0  1.0  1.0
```

## Specification of keywords

Although the keywords above are the same as for the case of *FermiLoop*, for user's convenience the specification of each keyword is explained with emphasis on *GridCalc* below. Note that the behavier of the keyword 'Enery.Range' is different from that in *FermiLoop*.

**Filename.scfout**
Specify the name of the scfout file which will be read by 'kSpin'.

**Filename.outdata**
Specify a name for output files. This keyword corresponds to the keyword 'System.Name' for OpenMX calculations.

**Calc.Type**
Choose either *FermiLoop*, *GridCalc*, *BandDispersion*, or *MulPOnly*. The default setting is *MulPOnly*. Here we choose *GridCalc* for the exercise.

**Energy.Range**
Specify the two different values for an energy range for which GridCalc should search bands. The unit is in eV. This keyword should be valid when the keyword 'Calc.Bandbyband' is 'OFF' (cf. Calc.Bandbyband). The default is '-0.5 0.5' (i.e. the range is [-0.5, 0.5]).

**Search.kCentral**
Specify a set of three values for the central k-point around which *GridCalc* should search k-points on the specified energy range (i.e. values for the keyword 'Energy.Range'). The notation to specify the k-point follows the keyword 'Band.kpath'. The default is '0.0 0.0 0.0' (i.e. Γ-point).

**Calc.Type.3mesh**
Specify a plane on which the spin texture is calculated. Set a value '1', '2', and '3', for the case of $k_ak_b$-, $k_bk_c$-, and $k_ck_a$-planes, respectively. The default is '1'.

**kRange.3mesh**
Specify two values for a two-dimensional domain in the reciprocal space where k-points should be calculated. For example, if the value for the keyword 'Calc.Type.3mesh' is '1' ($k_ak_b$-plane), values '0.2 0.3' specifies a domain: $-0.2 \leq k_a \leq 0.2$, $-0.3 \leq k_b \leq 0.3$. The notation of the k-points follows the keyword 'Band.kpath'. The default is '0.5 0.5' (i.e. the whole of the first brillouin zone).

**k-plane.1stStep**
Specify the number of grid points to divide the domain by the keyword 'kRange.3mesh' for the first search. For example, if the value for the keyword 'Calc.Type.3mesh' is '1' ($k_ak_b$-plane), values '2 3' specifies the number of grid points as follows: 2 for $k_a$-axis, 3 for $k_b$-axis. The default is '2 2'.

**Calc.Bandbyband**
Specify if *GridCalc* should calculate given bands (ON) or not (OFF). (cf. Calc.BandMin; Calc.BandMax). The default is 'OFF'.

**Calc.BandMin**
Set the lower limit on the range for bands to calculate by specifying the band index. This keyword is valid when the value for the keyword 'Calc.Bandbyband' is 'ON' (cf. Calc.Bandbyband; Calc.BandMax). You can see band indices using OMXTool [146] or 53.4 BandDispersion.

**Calc.BandMax**
Set the upper limit on the range for bands to calculate by specifying the band index. This keyword is valid when the value for the keyword 'Calc.Bandbyband' is 'ON' (cf. Calc.Bandbyband; Calc.BandMin). You can see band indices using OMXTool [146] or 53.4 BandDispersion.

**MulP.Vec.Scale**
Specify a scale to draw vectors expressing the spin texture. For example, values '0.1 0.2 0.3' specifies the scale as follows: 0.1 for x-axis, 0.2 for y-axis, 0.3 for z-axis. This keyword affects only 'XXXXX.Pxyz_YY' (XXXXX = the value for the keyword 'Filename.outdata'; YY = the band index). The default is '1.0 1.0 1.0'.

## Calculation

The spin texture/k-space spin density matrix are calculated by a post-processing code 'kSpin' in the directory 'work'. Then, please move to the directory 'work', and perform a calculation as follows:

```
% ./kSpin Au111Surface_GC.dat
```

or for the MPI calculation, for example, the case with 4 MPI processes

```
% mpirun -np 4 ./kSpin Au111Surface_GC.dat
```

As the calculation proceeds, you may see the following standard output:

```
*****************************************************************
*****************************************************************
 kSpin:
 code for evaluating spin related properties
 in momentum space of solid state materials.
 Copyright (C), 2019,
 Hiroki Kotaka, Naoya Yamaguchi and Fumiyuki Ishii.
 This software includes the work that is distributed
 in version 3 of the GPL (GPLv3).

 Please cite the following article:
 H. Kotaka, F. Ishii, and M. Saito,
 Jpn. J. Appl. Phys. 52, 035204 (2013).
 DOI: 10.7567/JJAP.52.035204.
*****************************************************************
*****************************************************************
```

```
Input filename is "Au111Surface.scfout"


Start "GridCalc" Calculation (4).

############ ORBITAL DATA ##################
ClaOrb_MAX[0]:   2
ClaOrb_MAX[1]:   8
Total Band (2*n): 124
Central (  0.000000   0.000000   0.000000)
##########################################

############ EIGEN VALUE ###################
The number of BANDs    8 (  49->  56)
Total MulP data:1568
##########################################

############ CALC TIME #####################
  Total Calculation Time:  3.656405 (s)
        Eigen Value Calc:  3.498849 (s)
##########################################
############ CALC TIME #####################
  Total Calculation Time:  3.670708 (s)
##########################################
```

When the calculation is completed normally as shown above, you can find the following output files in the directory 'work':

```
Au111Surface_GC.EigenMap_49
Au111Surface_GC.Pxyz_49
Au111Surface_GC.plotexample_49
Au111Surface_GC.EigenMap_50
Au111Surface_GC.Pxyz_50
Au111Surface_GC.plotexample_50
Au111Surface_GC.EigenMap_51
Au111Surface_GC.Pxyz_51
Au111Surface_GC.plotexample_51
Au111Surface_GC.EigenMap_52
Au111Surface_GC.Pxyz_52
Au111Surface_GC.plotexample_52
Au111Surface_GC.EigenMap_53
Au111Surface_GC.Pxyz_53
Au111Surface_GC.plotexample_53
Au111Surface_GC.EigenMap_54
Au111Surface_GC.Pxyz_54
```

```
Au111Surface_GC.plotexample_54
Au111Surface_GC.EigenMap_55
Au111Surface_GC.Pxyz_55
Au111Surface_GC.plotexample_55
Au111Surface_GC.EigenMap_56
Au111Surface_GC.Pxyz_56
Au111Surface_GC.plotexample_56
Au111Surface_GC.AtomMulP
Au111Surface_GC.MulP_s
Au111Surface_GC.MulP_p
Au111Surface_GC.MulP_p1
Au111Surface_GC.MulP_p2
Au111Surface_GC.MulP_p3
Au111Surface_GC.MulP_d
Au111Surface_GC.MulP_d1
Au111Surface_GC.MulP_d2
Au111Surface_GC.MulP_d3
Au111Surface_GC.MulP_d4
Au111Surface_GC.MulP_d5
Au111Surface_GC.atominfo
temporal_12345.input
```

As an example, by executing the following command, you can obtain figures of the spin texture for the Rashba spin splitting in the Au(111) surface as shown in Figs. 66(a) and (b). which exhibit the typical Rashba-type spin texture.

```
% gnuplot Au111Surface_GC.plotexample_55
% gnuplot Au111Surface_GC.plotexample_56
```

## Output files

The content of each output file is explained below:

**EigenMap_YY file**
This file stores a grid data for the band with the band index YY. The first, second, and third columns correspond to the $k_x$, $k_y$, and $k_z$ components of the k-points in units of Bohr$^{-1}$, respectively. The fourth column corresponds to the energy in units of eV.

**Pxyz_YY file**
This file stores data of the expectation value of the Pauli matrices vectors for each k-point stored in the FermiSurf_YY file. The first, second, and third columns correspond to the $k_x$, $k_y$, and $k_z$ component of the k-points in units of Bohr$^{-1}$, respectively. The fourth, fifth, and sixth columns correspond to the expectation value of the $\sigma_x$, $\sigma_y$, and $\sigma_z$ in units of the Bohr magneton, respectively.

**AtomMulP file**

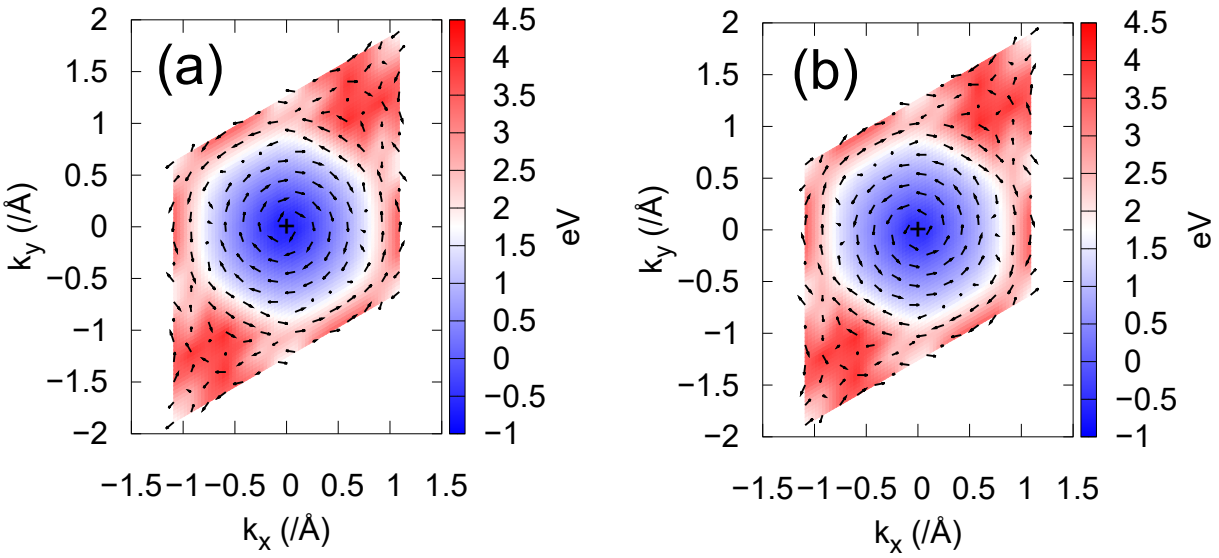



Figure 66: Spin texture for the band with (a) the band indice 55 and (b) 56 in the Rashba spin splitting around Γ-point at the Au(111) surface. The vectors represent the expectation values of the Pauli matrices vectors at each k-point stored in a Pxyz_YY file 'Au111Surface_GC.Pxyz_55' and 'Au111Surface_GC.Pxyz_56', respectively. The color maps represent the height in energy at each k-point stored in an EigenMap_YY file 'Au111Surface_GC.EigenMap_55' and 'Au111Surface_GC.EigenMap_56', respectively. The central cross point corresponds to Γ-point. You can get thes figures by plotexample_YY files 'Au111Surface_GC.plotexample_55' and 'Au111Surface_GC.plotexample_56'.

This file stores data of the k-space spin density matrix resolved to the atomic contribution, which can be analyzed by 53.6 *MulPCalc*.

**MulP_xx file**
This file stores data of the xx-component of the k-space spin density matrix resolved to the atomic contribution, and can be analyzed by 53.6 *MulPCalc*.

**plotexample file**
This file supplies an example of gnuplot scripts.

**atominfo file**
This file supplies information of lattice vectors and PAOs.

**temporal_12345.input**
This file is a copy of the input file stored in the scfout file.

## (Optional) Analysis of the k-space spin density matrix

*MulPCalc* can extract data to analyze the k-space spin density matrix from AtomMulP files or MulP_xx files. You can find an example of input files for *MulPCalc* at the bottom of an input file 'Au111Surface_GC.dat' and proceed to the analysis of the k-space spin density matrix after the above

calculation as follows:

```
% ./MulPCalc Au111Surface_GC.dat
```

For more information of *MulPCalc*, see also the subsection of 53.6 *MulPCalc*.

## 53.4 BandDispersion: Calculation on the band dispersion relation

'kSpin' has four ways to calculate the k-space spin density matrix, which can be specified by a keyword 'Calc.Type'. Here we introduce *BandDispersion* among the four ways. *BandDispersion* can calculate the k-space spin density matrix resolved to each atom on the band dispersion relation within a user-specified energy range. First, the calculation of the band dispersion using *BandDispersion* is illustrated here with a simple model of an Au(111) surface. Then, you can calculate spin textures using 53.6 *MulPCalc*.

After a calculation of OpenMX with an input file 'Au111Surface_BD.dat' stored in the directory 'work', you may try to run 'kSpin'. The keywords relevant to the executation of kSpin can be found at the bottom of the input file 'Au111Surface_BD.dat' stored in the directory 'work' as shown below:

### List of keywords relevant to kSpin

```
Filename.scfout      Au111Surface.scfout
Filename.outdata     Au111Surface_BD
Calc.Type            BandDispersion    # FermiLoop, GridCalc,
                                         BandDispersion, or MulPOnly
                                         default: MulPOnly
Energy.Range        -1.0  1.0          # eV; default: 0.0  0.0
Band.Nkpath               2
<Band.kpath
  135  0.0  0.500000  0.000000    0.0   0.000000  0.000000     M  G
  135  0.0  0.000000  0.000000    0.0  -0.500000  0.000000     G -M
Band.kpath>
```

### Specification of keywords

The specification of each keyword is explained below:

**Filename.scfout**
Specify the name of the scfout file which will be read by 'kSpin'.

**Filename.outdata**
Specify a name for output files. This keyword corresponds to the keyword 'System.Name' for OpenMX calculations.

**Calc.Type**
Choose either *FermiLoop*, *GridCalc*, *BandDispersion*, or *MulPOnly*. The default setting is *MulPOnly*. Here we choose *BandDispersion* for the exercise.

**Energy.Range**

Specify the two different values for an energy range for which BandDispersion should search bands. The unit is in eV. The default is '-0.5 0.5' (i.e. the range is [-0.5, 0.5]).

**Band.Nkpath**
Specify the number of k-paths along which*BandDispersion* should calculate the band dispersion. The notation is the same as that in the conventional calculation.

**Band.kpath**
Specify k-paths along which *BandDispersion* should calculate the band dispersion. The notation is the same as that in the conventional calculation.

## Calculation

The k-space spin density matrix resolved to each atom is calculated by a post-processing code 'kSpin' in the directory 'work'. Please move to the directory 'work', and perform a calculation as follows:

```
% ./kSpin Au111Surface_BD.dat
```

or for the MPI calculation, for example, the case with 4 MPI processes

```
% mpirun -np 4 ./kSpin Au111Surface_BD.dat
```

As the calculation proceeds, you may see the following standard output:

```
*******************************************************************
*******************************************************************
 kSpin:
 code for evaluating spin related properties
 in momentum space of solid state materials.
 Copyright (C), 2019,
 Hiroki Kotaka, Naoya Yamaguchi and Fumiyuki Ishii.
 This software includes the work that is distributed
 in version 3 of the GPL (GPLv3).

 Please cite the following article:
 H. Kotaka, F. Ishii, and M. Saito,
 Jpn. J. Appl. Phys. 52, 035204 (2013).
 DOI: 10.7567/JJAP.52.035204.
*******************************************************************
*******************************************************************


Input filename is "Au111Surface.scfout"


Start "BandDispersion" Calculation (3).
line_Nk[1]:  0.665829
```

```
( 0.000000,  0.500000,  0.000000 ) -> ( 0.000000,  0.000000,  0.000000 )
line_Nk[2]:  1.331658
( 0.000000,  0.000000,  0.000000 ) -> ( 0.000000, -0.500000,  0.000000 )
############ ORBITAL DATA ##################
ClaOrb_MAX[0]:   2
ClaOrb_MAX[1]:   8
Total Band (2*n): 124
##########################################
Band.Nkpath: 2
135 ( 0.000000,  0.500000,  0.000000 ) >>> ( 0.000000,  0.000000,  0.000000 ) M G
135 ( 0.000000,  0.000000,  0.000000 ) >>> ( 0.000000, -0.500000,  0.000000 ) G -M
l_min:  49   l_max:  56   l_cal:   8
Au111Surface_BD.Band49_1
Au111Surface_BD.Band50_1
Au111Surface_BD.Band51_1
Au111Surface_BD.Band52_1
Au111Surface_BD.Band53_1
Au111Surface_BD.Band54_1
Au111Surface_BD.Band55_1
Au111Surface_BD.Band56_1
l_min:  49   l_max:  56   l_cal:   8
Au111Surface_BD.Band49_2
Au111Surface_BD.Band50_2
Au111Surface_BD.Band51_2
Au111Surface_BD.Band52_2
Au111Surface_BD.Band53_2
Au111Surface_BD.Band54_2
Au111Surface_BD.Band55_2
Au111Surface_BD.Band56_2
##########################################
Total MulP data:2176
############ CALC TIME ####################
  Total Calculation Time:  4.772406 (s)
##########################################
```

When the calculation is completed normally as shown above, you can find the following output files in the directory 'work':

```
Au111Surface_BD.BAND
Au111Surface_BD.Band49_1
Au111Surface_BD.Band50_1
Au111Surface_BD.Band51_1
Au111Surface_BD.Band52_1
Au111Surface_BD.Band53_1
Au111Surface_BD.Band54_1
```

```
Au111Surface_BD.Band55_1
Au111Surface_BD.Band56_1
Au111Surface_BD.Band49_2
Au111Surface_BD.Band50_2
Au111Surface_BD.Band51_2
Au111Surface_BD.Band52_2
Au111Surface_BD.Band53_2
Au111Surface_BD.Band54_2
Au111Surface_BD.Band55_2
Au111Surface_BD.Band56_2
Au111Surface_BD.AMulPBand
Au111Surface_BD.AMulPBand_s
Au111Surface_BD.AMulPBand_p
Au111Surface_BD.AMulPBand_p1
Au111Surface_BD.AMulPBand_p2
Au111Surface_BD.AMulPBand_p3
Au111Surface_BD.AMulPBand_d
Au111Surface_BD.AMulPBand_d1
Au111Surface_BD.AMulPBand_d2
Au111Surface_BD.AMulPBand_d3
Au111Surface_BD.AMulPBand_d4
Au111Surface_BD.AMulPBand_d5
Au111Surface_BD.plotexample
Au111Surface_BD.atominfo
temporal_12345.input
```

As an example, by executing the following command, you can obtain a figure of the band dispersion for the Rashba spin splitting in the Au(111) surface as shown in Fig. 67(a).

```
% gnuplot Au111Surface_BD.plotexample
```

In order to obtain information of spin textures on the band dispersion, you may try to analyze by 53.6 *MulPCalc*.

## Output files

The content of each output file is explained below:

**BAND file**
This file stores data for the band dispersion. The first and second columns correspond to the distance for the k-points along each k-path (in units of $\mathrm{Bohr}^{-1}$) and the energy for each of them (in units of eV), respectively.

**Band_YY_Z file**
This file stores data for the band dispersion of each branch with the band index YY and k-path index Z. The notation of contents of this file is the same as the BAND file.

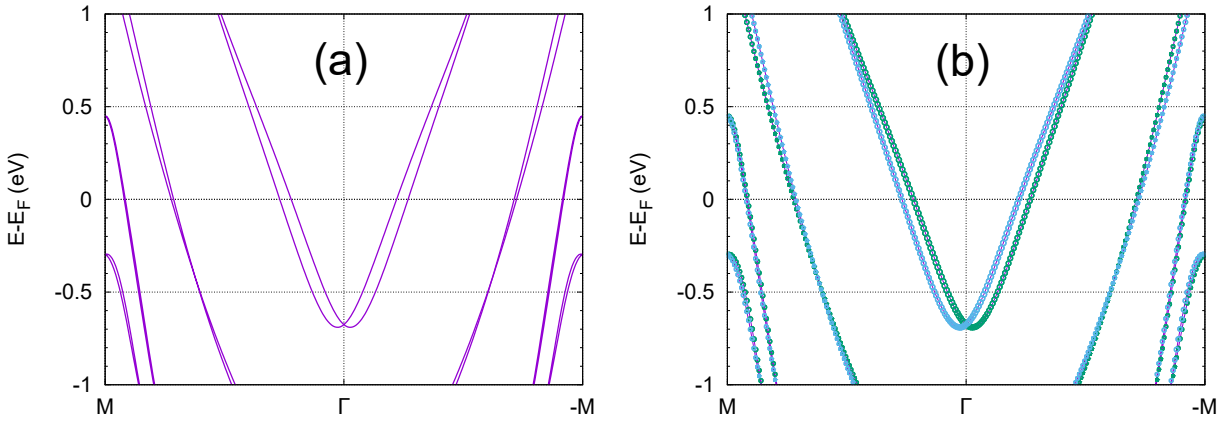



Figure 67: (a) Band dispersion for the Rashba spin splitting at the Au(111) surface stored in a BAND file 'Au111Surface_BD.BAND'. You can get this figure by a plotexample file 'Au111Surface_BD.plotexample'. (b) k-space spin density matrix resolved to each atom for the Rashba spin splitting at the Au(111) surface. The circles represent the difference between the number of electrons with $\alpha$- and $\beta$-spin at each k-point, and the radius and color reflect its magnitude and sign, respectively, stored in a MulPop file 'Au111Surface_BD_MC.MulPop'. You can get this figure by a plotexample file 'Au111Surface_BD_MC.plotexample'.

**AMulPBand file**
This file stores data of the k-space spin density matrix resolved to each atom, and can be analyzed by 53.6 *MulPCalc*.

**AMulPBand_xx file**
This file stores data of the xx-component of the k-space spin density matrix resolved to each atom, and can be analyzed by 53.6 *MulPCalc*.

**plotexample file**
This file supplies an example of gnuplot scripts.

**atominfo file**
This file supplies information of lattice vectors and PAOs.

**temporal_12345.input**
This file is a copy of the input file stored in the scfout file.

## Analysis of the k-space spin density matrix resolved to each atom

*MulPCalc* can extract data to analyze the k-space spin density matrix resolved to each atom from AMulPBand files or AMulPBand_xx files. The executable file can be obtained by compilation in the directory 'source' as follows:

```
% make MulPCalc
```

After the successful compilation, you can find the executable file 'MulPCalc' in the directory 'work'.

Let us analyze the k-space spin density matrix resolved to the atomic contribution on the band dispersion for the Rashba spin splitting in the Au(111) surface. First, add the following keywords and values into the input file 'Au111Surface_BD.dat', for example:

```
Filename.atomMulP Au111Surface_BD.AMulPBand # default: default
Filename.xyzdata   Au111Surface_BD_MC       # default: default
Num.of.Extract.Atom        3                # default: 1
Extract.Atom               1 2 3            # default: 1 2 ... (Num.of.Extract.Atom)
```

After the above calculation, you can analyze the k-space spin density matrix resolved to the atomic contribution as follows:

```
% ./MulPCalc Au111Surface_BD.dat
```

In addtion, you can adjust data for making better figures by the following keywords and values:

```
MulP.Vec.Scale       0.1  0.1  0.1          # default: 1.0  1.0  1.0
Data.Reduction             1                # default: 1
```

After executing *MulPCalc*, you can find the following output files in the directory 'work'.

```
Au111Surface_BD_MC.MulPop
Au111Surface_BD_MC.MulPop49
Au111Surface_BD_MC.MulPop50
Au111Surface_BD_MC.MulPop51
Au111Surface_BD_MC.MulPop52
Au111Surface_BD_MC.MulPop53
Au111Surface_BD_MC.MulPop54
Au111Surface_BD_MC.MulPop55
Au111Surface_BD_MC.MulPop56
Au111Surface_BD_MC.plotexample
```

As an example, by executing the following command, you can obtain a figure of the band dispersion for the Rashba spin splitting in the Au(111) surface with the k-space spin density matrix resolved to the atom contribution as shown in Fig. 67(b).

```
% gnuplot Au111Surface_BD_MC.plotexample
```

For more information of *MulPCalc*, see also the subsection of 53.6 *MulPCalc*.

### 53.5 MulPOnly: Calculation on user-specified k-points

'kSpin' has four ways to calculate the k-space spin density matrix, which can be specified by a keyword 'Calc.Type'. Here we introduce *MulPOnly* among the four ways. *MulPOnly* can calculate the k-space spin density matrix on user-specified k-points for user-specified bands. The calculation of spin texture using *MulPOnly* and *MulPCalc* is illustrated here with a simple model of an Au(111) surface.

After a calculation of OpenMX with an input file 'Au111Surface_MO.dat' stored in the directory 'work', you may try to run 'kSpin'. The keywords relevant to the executation of kSpin can be found at the bottom of the input file 'Au111Surface_MO.dat' stored in the directory 'work' as shown below:

**List of keywords relevant to kSpin**

```
Filename.scfout       Au111Surface.scfout
Filename.outdata      Au111Surface_MO
Calc.Type             MulPOnly              # FermiLoop, GridCalc,
                                              BandDispersion, or MulPOnly
                                              default: MulPOnly
Filename.kpointdata   kpoint.in
```

In addition, another input file, whose file name is specified by the keyword 'Filename.kpointdata', is required to specify k-points on which the k-space spin density matrix is calculated by *MulPOnly*. For the exercise, the file 'kpoint.in' is stored in the directory 'work'. The notation follows the following rule: In the first row, you need to specify the number of k-points where *MulPOnly* will calculate the k-space spin density matrix. Then, specify a k-point (the first, second, and third columns correspond to $k_x$, $k_y$, and $k_z$, respectively, in Bohr$^{-1}$.) and a band index (the fourth column) for it. The file 'kpoint.in' stores information of the k-points on the circle around the Γ-point in the reciprocal space. For example, you can prepare this file as follows:

```
rm -f kpoint.in && awk '{for (i=0; i<$1; i++){printf "%17.14f %17.14f %17.14f %d\n", 0, $2*cos(2*atan2(0, -1)/$1*i),
$2*sin(2*atan2(0, -1)/$1*i), $3 >> "buffer"}; sum+=$1; if (!$1) {print sum; exit}}' - > N.in && cat N.in buffer >> kpoint.in && rm N.in buffer
```

After that, if you give values as follows:

```
40 0.18 55          <- # of k-points; radius; band index
40 0.15 56          <- # of k-points; radius; band index
0                   <- exit
```

Then, you will get 'kpoint.in' and you can see the content of 'kpoint.in' as follows:

```
% cat kpoint.in
80
0.00000000000000 0.18000000000000 0.00000000000000 55
0.00000000000000 0.17778390130712 0.02815820370724 55
0.00000000000000 0.17119017293313 0.05562305898749 55
0.00000000000000 0.16038117435391 0.08171828995312 55
0.00000000000000 0.14562305898749 0.10580134541265 55
0.00000000000000 0.12727922061358 0.12727922061358 55
0.00000000000000 0.10580134541265 0.14562305898749 55
0.00000000000000 0.08171828995312 0.16038117435391 55
0.00000000000000 0.05562305898749 0.17119017293313 55
0.00000000000000 0.02815820370724 0.17778390130712 55
0.00000000000000 0.00000000000000 0.18000000000000 55
0.00000000000000 -0.02815820370724 0.17778390130712 55
...
..
```

## Specification of keywords

The specification of each keyword is explained below:

**Filename.scfout**
Specify the name of the scfout file which will be read by 'kSpin'.

**Filename.outdata**
Specify a name for output files. This keyword corresponds to the keyword 'System.Name' for OpenMX calculations.

**Calc.Type**
Choose either *FermiLoop*, *GridCalc*, *BandDispersion*, or *MulPOnly*. The default setting is *MulPOnly*. Here we choose *MulPOnly* for the exercise.

**Filename.kpointdata**
Specify the name of an data file of k-points and band indices.

## Calculation

The k-space spin density matrix resolved to each atom is calculated by a post-processing code 'kSpin' in the directory 'work'. Please move to the directory 'work', and perform a calculation as follows:

```
% ./kSpin Au111Surface_MO.dat
```

or for the MPI calculation, for example, the case with 4 MPI processes

```
% mpirun -np 4 ./kSpin Au111Surface_MO.dat
```

As the calculation proceeds, you may see the following standard output:

```
***********************************************************************
***********************************************************************
 kSpin:
 code for evaluating spin related properties
 in momentum space of solid state materials.
 Copyright (C), 2019,
 Hiroki Kotaka, Naoya Yamaguchi and Fumiyuki Ishii.
 This software includes the work that is distributed
 in version 3 of the GPL (GPLv3).

 Please cite the following article:
 H. Kotaka, F. Ishii, and M. Saito,
 Jpn. J. Appl. Phys. 52, 035204 (2013).
 DOI: 10.7567/JJAP.52.035204.
***********************************************************************
***********************************************************************


Input filename is "Au111Surface.scfout"


Start "MulPOnly" Calculation (6).
```

```
########### ORBITAL DATA ##################
ClaOrb_MAX[0]:   2
ClaOrb_MAX[1]:   8
Total Band (2*n): 124
##########################################

############ CALC TIME ####################
  Total Calculation Time:  0.642631 (s)
##########################################
############ CALC TIME ####################
  Total Calculation Time:  0.660566 (s)
##########################################
```

When the calculation is completed normally as shown above, you can find the following output files in the directory 'work':

```
Au111Surface_MO.AtomMulP
Au111Surface_MO.MulP_s
Au111Surface_MO.MulP_p
Au111Surface_MO.MulP_p1
Au111Surface_MO.MulP_p2
Au111Surface_MO.MulP_p3
Au111Surface_MO.MulP_d
Au111Surface_MO.MulP_d1
Au111Surface_MO.MulP_d2
Au111Surface_MO.MulP_d3
Au111Surface_MO.MulP_d4
Au111Surface_MO.MulP_d5
Au111Surface_MO.atominfo
temporal_12345.input
```

## Output files

The content of each output file is explained below:

**AtomMulP file**
This file stores data of the k-space spin density matrix resolved to each atom, and can be analyzed by 53.6 *MulPCalc*.

**MulP_xx file**
This file stores data of the xx-component of the k-space spin density matrix resolved to each atom, and can be analyzed by 53.6 *MulPCalc*.

**atominfo file**

This file supplies information of lattice vectors and PAOs.

**temporal_12345.input**
This file is a copy of the input file stored in the scfout file.

## Analysis of the k-space spin density matrix resolved to each atom

*MulPCalc* can extract data to analyze the k-space spin density matrix resolved to each atom from AMulPBand files or AMulPBand_xx files. The executable file can be obtained by compilation in the directory 'source' as follows:

```
% make MulPCalc
```

After the successful compilation, you can find the executable file 'MulPCalc' in the directory 'work'. Let us analyze spin textures for the Rashba spin splitting in the Au(111) surface. First, add the following keywords and values into the input file 'Au111Surface_MO.dat', for example:

```
Filename.atomMulP Au111Surface_MO.AMulPBand   # default: default
Filename.xyzdata   Au111Surface_MO_MC         # default: default
Num.of.Extract.Atom        3                  # default: 1
Extract.Atom               1 2 3              # default: 1 2 ... (Num.of.Extract.Atom)
Calc.Type.3mesh            2                  # default: 1
```

And, after the above calculation, you can analyze spin textures as follows:

```
% ./MulPCalc Au111Surface_MO.dat
```

In addtion, you can adjust data for making better figures by the following keywords and values:

```
MulP.Vec.Scale       0.1  0.1  0.1          # default: 1.0  1.0  1.0
Data.Reduction             1                # default: 1
```

After executing *MulPCalc*, you can find the following output files in the directory 'work'.

```
Au111Surface_MO_MC.MulPop
Au111Surface_MO_MC.MulPop55
Au111Surface_MO_MC.MulPop56
Au111Surface_MO_MC.plotexample
```

As an example, by executing the following command, you can obtain a figure of spin textures for the Rashba spin splitting in the Au(111) surface as shown in Fig. 68.

```
% gnuplot Au111Surface_MO_MC.plotexample
```

For more information of *MulPCalc*, see also the subsection of 53.6 *MulPCalc*.

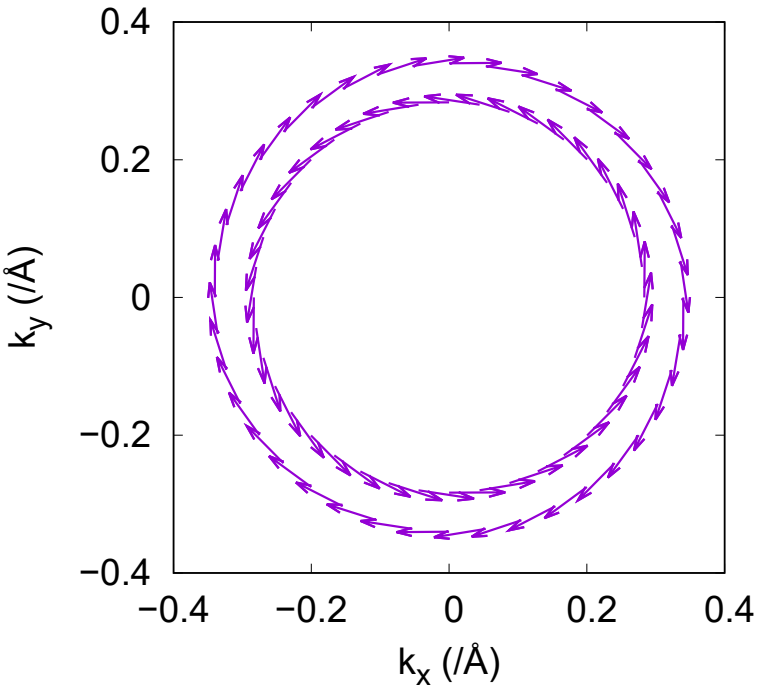



Figure 68: Spin textures for the Rashba spin splitting around Γ-point at the Au(111) surface. The outer and inner ones correspond to the band indices 55 and 56. The vectors represent the expectation values of the Pauli matrices vectors at each k-point stored in 'Au111Surface_MO_MC.MulPop'. You can get this figure by a plotexample file 'Au111Surface_MO_MC.plotexample'

### 53.6 MulPCalc: k-space spin density matrix resolved to each atom

*MulPCalc* can extract data to analyze the Mulliken population from AtomMulP files, MulP_xx files, AMulPBand files or AMulPBand_xx files obtained by kSpin. The executable file can be obtained by compilation in the directory 'source' as follows:

```
% make MulPCalc
```

After the successful compilation, you will find the executable file 'MulPCalc' in the directory 'work'. Also, you find an example of an input file 'SiC_Primitive_BD.dat' in the directory 'work'. After the SCF calculation, please perform a *BandDispersion* calculation by 'kSpin' with the following settings:

```
Filename.scfout    sic_primitive.scfout # default: default
Filename.outdata   sic_primitive_BD     # default: default
Calc.Type          BandDispersion       # default: MulPOnly
Energy.Range       -10.0  6.0           # eV; default: 0.0  0.0
```

Next, please execute *MulPCalc* as

```
% ./MulPCalc SiC_Primitive_BD.dat
```

with the following settings to extract data of $p_z$ orbitals on a carbon atom:

```
Filename.atomMulP sic_primitive_BD.AMulPBand_p3 # default: default
Filename.xyzdata   sic_primitive_BD_MC_C_p3     # default: default
Num.of.Extract.Atom        1                   # default: 1
Extract.Atom               1                   # default: 1 2 ... (Num.of.Extract.Atom)
```

In addtion, you can adjust data for making better figures by the following keywords and values:

```
MulP.Vec.Scale       0.1  0.1  0.1            # default: 1.0  1.0  1.0
Data.Reduction             2                  # default: 1
```

Figure 69 can be obtained by executing *MulPCalc* several times similarly and plotting extracted data of the eleventh, four, and fifth columns stored in MulPop files by using the circles style of gnuplot.

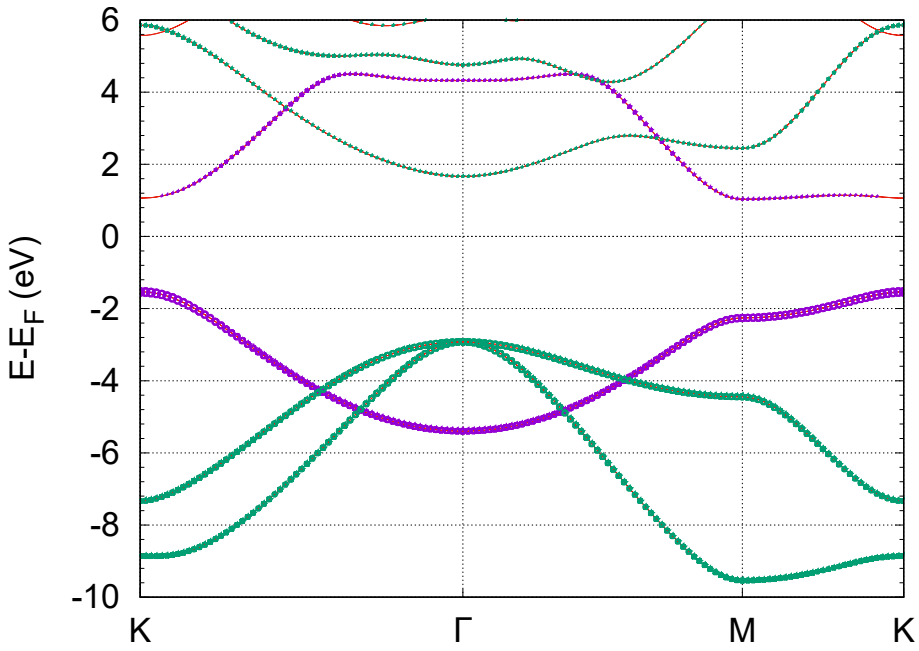



Figure 69: k-space spin density matrix resolved to each atom for the SiC primitive cell in a two-dimensional haneycomb structure without imperfection as explained in the subsection of 52.1 Analysis of band structures under the section of 52 Unfolding method for band structures. The green and purple circles represent the number of elecrons occupying $s$, $p_x$, and $p_y$ orbitals on a carbon atom, and that occupying $p_z$ orbitals on it, respectively. The radius reflects the magnitude of the number of electrons. Using *MulPCalc*, the data can be extracted from AMulPBand_xx files 'sic_primitive_BD.AMulPBand_s', 'sic_primitive_BD.AMulPBand_p1', 'sic_primitive_BD.AMulPBand_p2', or 'sic_primitive_BD.AMulPBand_p3' or 'sic_primitive_BD.AMulPBand_p'.

In addition, in the directory 'work', there is another example of an input file 'Au111Surface23_FL.dat' for a slab model with two clean Au(111) surfaces, which may take much more time than the other examples. You can try it similarly. After the SCF calculation and subsequent *FermiLoop* calculation, you may find the band dispersion and spin textures as shown in Figs. 70(a) and 70(b), respectively, where there are two pairs of Rashba bands, which are degenerated because one surface is equivalent to

the other. In this case, you may need to obtain the contribution of atoms in one surface by *MulPCalc*. By using *MulPCalc*, you can get it and find spin texture in one surface as shown in Fig. 70(c).

The specification of each keyword is explained below:

## List of keywords (common) relevant to MulPCalc

**Filename.atomMulP**
Specify the name of an AtomMulP file, a MulP_xx file, an AMulPBand file or an AMulPBand_xx file.

**Filename.xyzdata**
Specify a name for output files. This keyword corresponds to the keyword 'System.Name'.

**Num.of.Extract.Atom**
Specify the number of atoms for which MulPCalc should extract data of the Mulliken population. The default is '1'.

**Extract.Atom**
Specify atoms for which MulPCalc should extract data of the Mulliken population. The default is '1 2 ... (the value for the keyword 'Num.of.Extract.Atom')'.

**Data.Reduction**
Specify the number of k-points every which MulPCalc should extract data of the Mulliken population. This keyword is useful in thinning out k-points or reducing data size.

**MulP.Vec.Scale**
Specify a scale to draw vectors expressing spin textures. For example, values '0.1 0.2 0.3' specifies the scale as follows: 0.1 for x-axis, 0.2 for y-axis, 0.3 for z-axis. The default is '1.0 1.0 1.0'.

**Filename.outdata**
Keep the specification in kSpin.

**Calc.Type**
Keep the specification in kSpin.

## List of keywords (Calc.Type = FermiLoop or GridCalc) relevant to MulPCalc

**Search.kCentral**
Keep the specification in kSpin.

**Calc.Type.3mesh**
Keep the value in kSpin.

**Energy.Range**
Keep the value in kSpin.

## List of keywords (Calc.Type = BandDispersion) relevant to MulPCalc

**Energy.Range**
Keep the value in kSpin.

**Band.Nkpath**
Keep the value in kSpin.

**Band.kpath**
Keep the value in kSpin.

### List of keywords (Calc.Type = MulPOnly) relevant to MulPCalc

**Calc.Type.3mesh**
Specify a plane to calculate as follows: Set a value '1', '2', and , '3', for the case of $k_a k_b$-, $k_b k_c$-, and $k_c k_a$- planes, respectively. The default is '1'.

### Output files

After the calculation by 'MulPCal' is completed normally, you obtain the following output files in the working directory'.

**MulPop file**
This file stores data for each k-point. The first, second, and third columns correspond to the $k_x$, $k_y$, and $k_z$ components of the k-points in units of $\text{Å}^{-1}$, respectively. The fourth column corresponds to the energy in units of eV; The fifth, sixth, and seventh columns correspond to the number of electrons: Total, $\alpha$-spin, and $\beta$-spin, respectively. The eighth, nineth, and tenth columns correspond to the expectation value of the $\sigma_x$, $\sigma_y$, and $\sigma_z$ in units of the Bohr magneton, respectively. If the value for the keyword 'Calc.Type' is 'BandDispersion', the eleventh column corresponds to the distance for the k-points along each k-path in units of $\text{Bohr}^{-1}$.

**MulPop_YY file**
This file stores data for each k-point with the band index YY. The notation of contents of this file is the same as the MulPop file.

**plotexample file**
If the value for the keyword 'Calc.Type' is not 'GridCalc', this file supplies an example of gnuplot scripts.

**plotexample_YY file**
If the value for the keyword 'Calc.Type' is 'GridCalc', this file supplies an example of gnuplot scripts for the band with the band index YY.

## 53.7 MPI parallelization of kSpin

MPI parallelization is available for kSpin. More detailed information follows:

**FermiLoop**
Although MPI parallelization is done for k-points on a grid, the second search should be done in each domain picked out in the first search. Since the second search is done sequentially only in domains necessary for calculations, the maximum of the appropriate number of MPI processes is the product of the first and second values for the keyword 'k-plane.2ndStep'. Note that the second search should cost most of computation.

**GridCalc**
MPI parallelization is done for k-points on a grid so that the maximum of the appropriate number of MPI processes is the product of the first and second values for the keyword 'k-plane.1stStep'.

**BandDispersion**
MPI parallelization is done for k-points on every k-path so that the maximum of the appropriate number of MPI processes is the maximum number of k-points among k-paths.

**MulPOnly**
MPI parallelization is done for k-points so that the maximum of the appropriate number of MPI processes is the number of specified k-points.

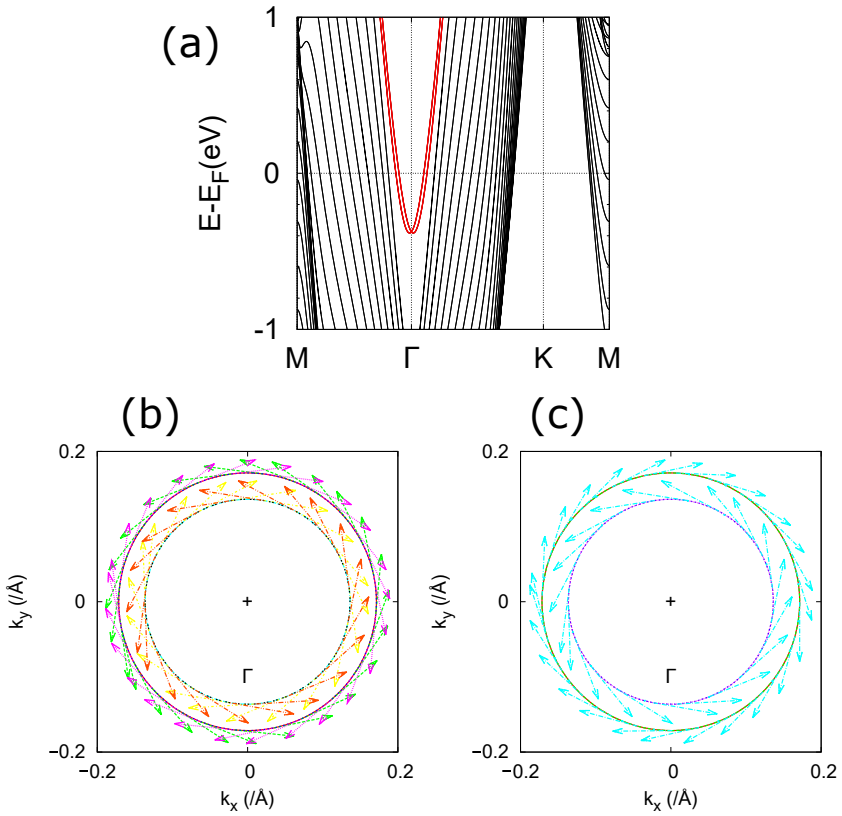



Figure 70: (a) Band dispersion for the Rashba spin splitting at both the Au(111) surfaces stored in a Band file 'Au111Surface23.Band' (See the section of 53.4 *BandDispersion*). The red curves show the Rashba bands, which correspond to band indices 413, 414, 415 and 416. This highlight of the Rashba bands can be reproduced by modifing a GNUBAND file 'Au111Surface23.GNUBAND' or using OMXTool [146]. Spin textures for the Rashba spin splitting around Γ-point at (b) both the Au(111) surfaces and (c) one of the Au(111) surfaces. The vectors represent the expectation values of the Pauli matrices vectors at each k-point stored in Pxyz_YY files 'Au111Surface23_FL.Pxyz_413', 'Au111Surface23_FL.Pxyz_414', 'Au111Surface23_FL.Pxyz_415' and 'Au111Surface23_FL.Pxyz_416', and 'Au111Surface23_FL_MC.Pxyz_413', 'Au111Surface23_FL_MC.Pxyz_414', 'Au111Surface23_FL_MC.Pxyz_415' and 'Au111Surface23_FL_MC.Pxyz_416', respectively. The closed curves represent the constant-energy level stored in FermiSurf_YY files 'Au111Surface23_FL.FermiSurf_413', 'Au111Surface23_FL.FermiSurf_414', 'Au111Surface23_FL.FermiSurf_415' and 'Au111Surface23_FL.FermiSurf_416', and 'Au111Surface23_FL_MC.FermiSurf_413', 'Au111Surface23_FL_MC.FermiSurf_414', 'Au111Surface23_FL_MC.FermiSurf_415' and 'Au111Surface23_FL_MC.FermiSurf_416', respectively. The central cross point corresponds to Γ-point. You can get these figures by plotexample files 'Au111Surface23_FL.plotexample' and 'Au111Surface23_FL_MC.plotexample', respectively.

## 54 Spin spiral calculations

Spin spiral calculations are supported for the non-collinear DFT without spin orbit coupling (SOC), which is based on the generalized Bloch theorem [79, 80, 86, 87]. It should be noted that the inclusion of SOC is not compatible with the functionality, since the SOC violates the spiral symmetry imposed by the generalized Bloch theorem. To acknowledge in any publications by using the functionality, the citation of the reference [79, 80] would be appreciated. To do the calculations, the following options are first set respectively

```
scf.SpinPolarization        NC        # On|Off|NC
scf.Generalized.Bloch       on        # On|Off, default=off
scf.SpinOrbit.Coupling      off       # On|Off, default=off
```

In the spiral structure there are two quantities to determine the spiral configuration which should be paid attention. The first one is the cone angle. Since the spiral structure is of two different kinds, i.e., the conical spiral ($0 < \theta < 90$) and the flat spiral ($\theta = 90$), where $\theta$ is the cone angle, $\theta$ should then be specified for the magnetic atoms. In OpenMX, the cone angle $\theta$ is defined as the orientation for the spin magnetic moment as shown in Fig. 71(a). You can find how to specify the cone (Euler) angle in section of 34 'Non-collinear DFT'. For example, the cone (Euler) angle can be specified as follows:

```
<Atoms.SpeciesAndCoordinates
 1  Fe  0.0    0.0    0.0     10.0    6.0   90.0   0.0   0.0   0.0  1 off
Atoms.SpeciesAndCoordinates>
```

In the example above, the flat spiral will be achieved by setting $\theta = 90$ degree as written in the 8th column. In addition, you can also specify the initial angle of $\varphi$ as described in the 9th column. For fixing the cone angle, the constraint scheme is also available as explained in section of 38 'Constraint DFT for non-collinear spin orientation'.

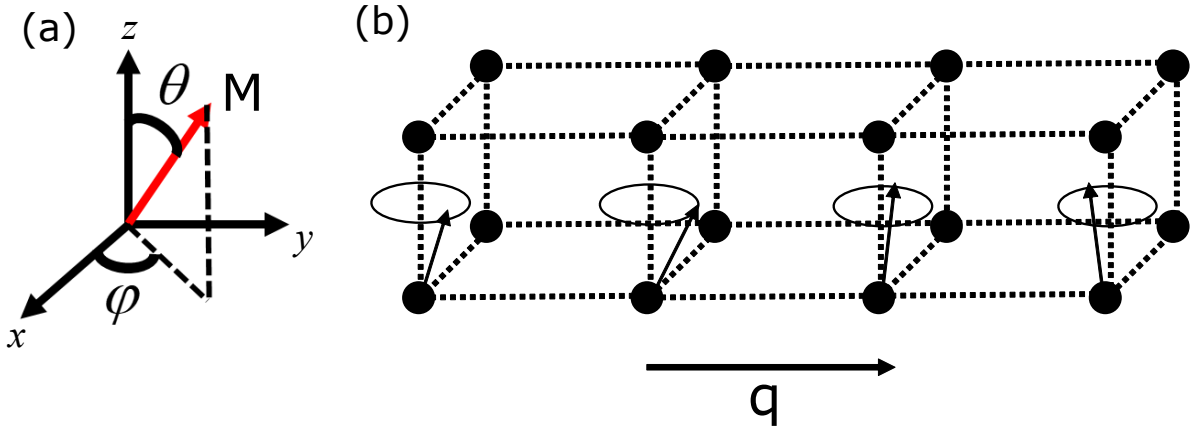



Figure 71: (a) Definition of cone (Euler) angles for the magnetic atom. (b) The magnetic moment of atom at site $i$ which is given by $M_i = M_i\{\cos(\mathbf{q}\cdot\mathbf{R}_i + \varphi_0)\sin(\theta_i) + \sin(\mathbf{q}\cdot\mathbf{R}_i + \varphi_0))\sin(\theta_i) + \cos(\theta_i)\}$.

The second one is the spiral vector **q** specified by

```
Spin.Spiral.Vector    0.0   0.0   0.0     # q1 q2 q3
```

In the above format, the spiral vector is specified by the fractional coordinates spanned by the reciprocal lattice vectors. The first, second, and third columns denote the components of spiral vector for the reciprocal vectors $\tilde{\mathbf{a}}_1$, $\tilde{\mathbf{a}}_2$, and $\tilde{\mathbf{a}}_3$, respectively. Then, the spin angle at atomic site $i$ is given by

$$M_i = M_i\{\cos(\mathbf{q}\cdot\mathbf{R}_i + \varphi_0)\sin(\theta_i) + \sin(\mathbf{q}\cdot\mathbf{R}_i + \varphi_0))\sin(\theta_i) + \cos(\theta_i)\}.$$

With the translation by the lattice vectors, the spin angle at each atomic site is rotated according to the equation.

As an example of the spiral calculation, the spiral ground state of the fcc Fe is provided in Fig. 72. We observe that the spiral ground state occurs at (0, 0, 0.6) and (0.2, 0, 1) defined in Cartesian coordinates in units of $2\pi/a$. Indeed, using LCPAO method, the spiral calculation needs the appropriate settings, such as the number of k points, number of orbitals, cutoff radius, and cutoff energy for reaching the convergence and reliable result, a similar discussion can be found in Ref. [87]. You can try to set those parameters to compare all of the results. In addition, although there are some available mixing schemes specified by the keywords 'scf.Mixing.Type', we strongly recommend RMM-DIISH as your choice. As an experience, this option can reach the convergence much faster than other choices for all tested systems, the detail explanations can be found in Sec. 13 'SCF convergence'.

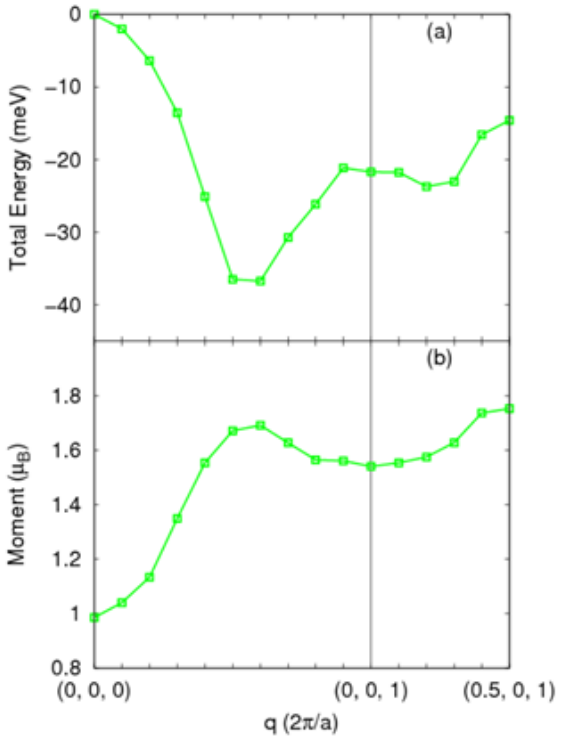



Figure 72: (a) Profiles of the total energy and (b) magnetic moment of the spin spiral calculation of the fcc Fe as a function of spiral vector **q** defined in the Cartesian coordinates. The input file used for the calculation is 'Fefcc-SpinSpiral.dat' in the directory 'work'.

# 55 Computing Chern number and Berry curvature by the Fukui-Hatsugai-Suzuki method

## 55.1 General

In OpenMX Ver. 3.9, a post-processing code 'calB' is supported to calculate the Chern number and Berry curvature of bands using overlap matrix elements between Kohn-Sham orbitals at neighboring k-points by the Fukui-Hatsugai-Suzuki method [81, 85]. The functionality is compatible with only the non-collinear calculations. To acknowledge in any publications by using the functionality, the citation of the reference [84] would be appreciated.

The Chern number is a topological invariant being an integer number, which characterizes the topology of bands for any materials. In systems having a finite Chern number $C$, the anomalous Hall conductivity defined by

$$\sigma_{xy} = -\frac{e^2}{h} C \ (C = 1, 2, 3, \cdots)$$

is induced. Using the Berry curvature $\mathbf{F}_n = \nabla \times \mathbf{A}_n,\ \mathbf{A}_n = -i\langle u_{n\mathbf{k}} | \frac{\partial}{\partial \mathbf{k}} | u_{n\mathbf{k}} \rangle$, the Chern number is defined as

$$C = \frac{1}{2\pi} \sum_n^{\text{occ.}} \int F_{nz} dk^2$$

In the Fukui-Hatsugai-Suzuki method [81], the overlap matrix $U$ defined by

$$U_{\Delta\mathbf{k}}(\mathbf{k}) = \det \langle u_n(\mathbf{k}) | u_m(\mathbf{k} + \Delta\mathbf{k}) \rangle$$

plays a central role to calculate the Berry connection $\mathbf{A}(\mathbf{k})$ and Berry curvature $F(\mathbf{k})$, which are defined by

$$\begin{aligned} \mathbf{A}(\mathbf{k}) &= \text{Im} \log U_{\Delta\mathbf{k}}(\mathbf{k}), \\ F(\mathbf{k}) &= \text{Im} \log U_{\Delta k_1}(\mathbf{k}) U_{\Delta k_2}(\mathbf{k} + \Delta k_1) U^{-1}_{\Delta k_1}(\mathbf{k} + \Delta k_2) U^{-1}_{\Delta k_2}(\mathbf{k}) \end{aligned}$$

As shown in Fig. 73, the Berry curvature can be calculated in each 'plaquette' (plaquette means meshed area in Brillouin zone) on a regular mesh introduced in the first Brillouin zone by the following formula:

$$F(\mathbf{k}) = \text{Im} \log U_{12} U_{23} U_{34} U_{41}$$

By summing up all the contributions of the contour integrals for the Berry curvature, one can calculate the Chern number as

$$C = \frac{1}{2\pi} \sum_{\mathbf{k}}^{\text{BZ}} F(\mathbf{k})$$

## 55.2 Example

As an example, we show in Fig. 74 the Berry curvature of graphene. Since the absolute magnitude of Berry curvature is approximately proportional to the square of inverse of bandgap, the large Berry

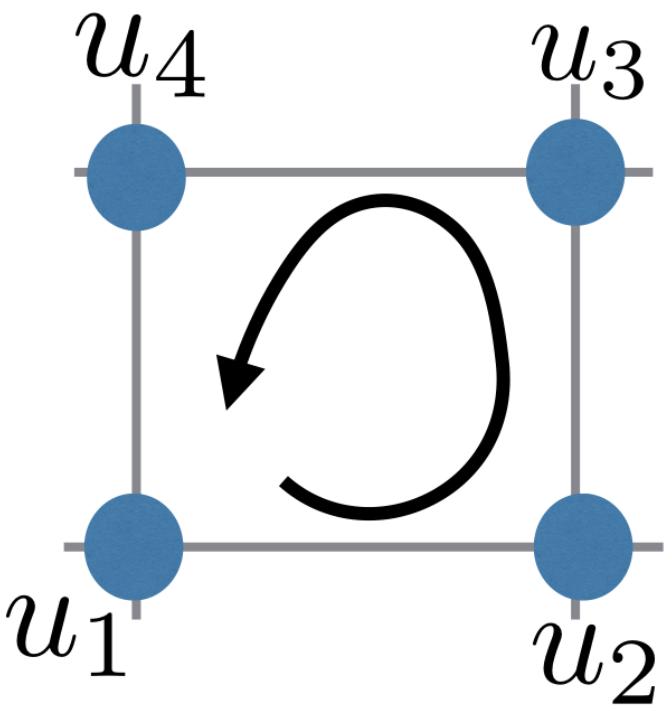



Figure 73: Computational method of the Berry curvature, where a contour integral is performed in each plaquette.

curvature can be seen around K and K' points, where the massive Dirac point appears if we include spin-orbit interaction.

Let us illustrate how the calculation can be performed by using an input file 'Graphene-Chern.dat' stored in the directory 'work'.

**SCF calculation**

When you perform the SCF calculation for graphene using an input file 'Graphene-Chern.dat' stored in the directory 'work', you need to switch on the keyword 'HS.fileout' as

```
HS.fileout      on      #on|off, default=off
```

After finishing the SCF calculation normally, we may obtain an out file 'Graphene-Chern.scfout'.

**Calculation of Chern number and Berry curvature**

Then, you can proceed for the calculation of the Chern number of Berry curvature using a post-processing code 'calB.c'. The compilation of the code can be done in the director 'source' as

```
% make calB
```

After the compilation is successfuly, you obtain the executable file 'calB'. Then, please copy it to the directory 'work', and execute it as follows:

```
% ./calB graphene.scfout
```

or

```
   % ./calB graphene.scfout < calB.in > calB.out
or
   % mpirun -np 4 ./calB graphene.scfout < calB.in > calB.out
```

### Parameters for calB

The input file 'calB.in' needs to be prepared as follows:

```
1
0
0 0 1
100 100
```

In the following we explain parameters for the calculation.

- In the first line, you need to specify whether the sum of the Berry curvature of all the bands is calculated, or the Berry curvature is calculated for a disentangled bands. If the former is selected, please set '1', while for the latter please set '0'. The latter setting is valid only if a set of bands are fully disentangled.
- In the second line, you need to specify how many bands are taken into account for the calculations, which are counted from the lowest band. If you specify '0', all the valence bands will be taken into account.
- In the third line, we specify planes to be calculated in the reciprocal space. For example, '0 0 1' corresponds to the specification of the $k_3 = 0$ plane.
- In the fourth line, you need to specify the number of mesh. In the example of graphene, the plane specified by the third line is discretized by a mesh of 100×100.

### Output files

After the calculation by 'calB', the following files are generated.

- BerryC*.dat

  The file stores the calculated Berry curvature and Chern number, where the '*' behind 'BerryC' in the file name is the number of bands to be included. In the example of graphene, '*' is 8. The unit of Berry curvature is $Å^{-2}$. The part of 'BerryC8.dat' for the graphene case is shown below:

```
#Mesh Number:100*100
#Band Number:8
#ChernNumber = 0.000000
#k1          k2           F (ang-2)
0.005000 0.005000 -1.59799534e-05
0.015000 0.005000 1.94455630e-05
0.025000 0.005000 3.58476883e-05
....
..
```

Using gnuplot, one can visualize the data as

```
% splot "BerryC8.dat" w l
```

Figure 74 shows the Berry curvature on the plane at $k_3 = 0$.

- temporal_12345.input

  This is a copy of input file, which was used for the SCF calculation, reconstructed from the scfout file.

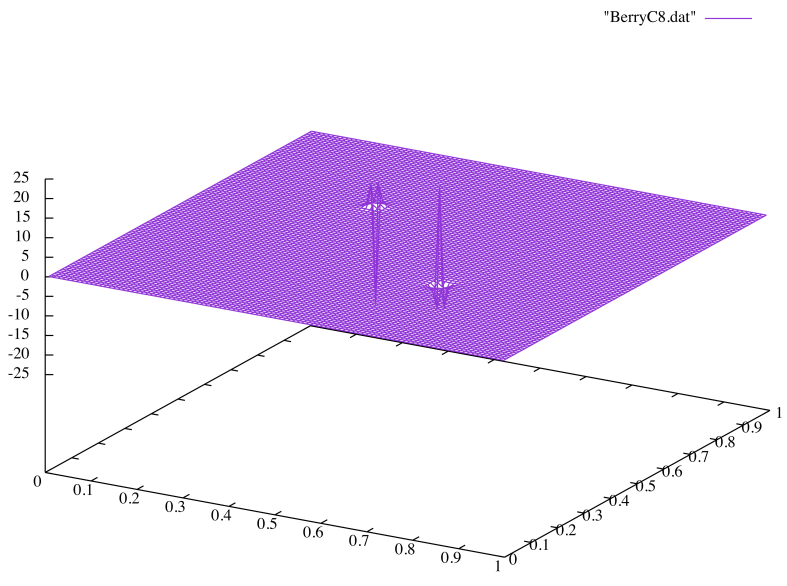



Figure 74: Berry curvature of graphene bands.

# 56 Computing $Z_2$ invariant by the Fukui-Hatsugai method

## 56.1 General

The $Z_2$ invariant of the system can be calculated with a method based on the Berry phase formalism proposed by Fukui and Hatsugai [81, 82]. The functionality is compatible with only the non-collinear calculations. Also, magnetic systems cannot be treated by the current implementation. To acknowledge in any publications by using the functionality, the citation of the reference [84] would be appreciated.

The $Z_2$ invariant is a topological invariant number being 0 or 1, which is defined on time reversal symmetric non-magnetic systems. $Z_2 = 1$ and $Z_2 = 0$ correspond to topological and trivial insulators, respectively. The $Z_2$ invariant is defined as

$$Z_2 = \frac{1}{2\pi}\sum_{n}^{\text{occ.}} \left( \int_{\partial B} \mathbf{A}_n \cdot d\mathbf{k} - \int_B F_{nz} dk^2 \right) \text{ (mod 2)}$$

where $\mathbf{A}_n = -i\langle u_{n\mathbf{k}} | \frac{\partial}{\partial \mathbf{k}} | u_{n\mathbf{k}} \rangle$ is called Berry connection, and $\mathbf{F}_n = \nabla \times \mathbf{A}_n$ is called Berry curvature. The integration range $B = [-\frac{\mathbf{G}_1}{2}, \frac{\mathbf{G}_1}{2}] \otimes [0, \frac{\mathbf{G}_2}{2}]$ is enough to consider only the half of Brillouin zone. This is because the system has the time-reversal symmetry, and thereby the topological invariant is defined on the half of Brillouin zone. For performing the integration, we use the overlap matrix $U$, proposed by Fukui, Hatsugai, and Suzuki [81, 82], defined by $U_{\Delta\mathbf{k}} = \det\langle u_n(\mathbf{k}) | u_m(\mathbf{k} + \Delta\mathbf{k}) \rangle$, and calculate the Berry connection and Berry curvature on every 'plaquette', which means meshed area in the Brillouin zone, as

$$\begin{aligned} A_{ab} &= \operatorname{Im} \log U_{ab}, \\ F &= \operatorname{Im} \log U_{12} U_{23} U_{34} U_{41}. \end{aligned}$$

Then, the integer-valued field $n(= 0, \pm 1)$ on every plaquette can be calculated by the following formula;

$$n(\mathbf{k}) = \frac{1}{2\pi}(A_{12} + A_{23} + A_{31} + A_{41} - F)$$

By summing up all the $n$ on the half Brillouin zone, and considering the modulo 2 of the summed value, we can obtain the $Z_2$ invariant. It should be noted that the $Z_2$ invariant is gauge independent, but the value of each $n$ is gauge dependent, which may vary depending on computational environment, compiler optimization level, and a tiny difference in the electronic structure. The details of computing $\mathbf{A}$ and $F$ is explained in Section of "Chern number and Berry curvature". Please refer it. Since the calculation of the $Z_2$ invariant is carried out by the contour integral on the half of Brillouin zone, it depends on arbitrariness of wave function's gauges. Therefore, we have to fix the gauges on the boundary of half Brillouin zone. As shown in Fig. 75, we consider the following three kinds of gauge fix on the boundary:

**Translational symmetry (red parts)**

$$|u_n(\mathbf{k} + \frac{\mathbf{G}_1}{2})\rangle = e^{i\mathbf{G}_1 \cdot \mathbf{r}} |u_n(\mathbf{k} - \frac{\mathbf{G}_1}{2})\rangle$$

**Time-reversal symmetry (blue parts)**

$$|u_n(\mathbf{k})\rangle = \Theta |u_n(-\mathbf{k})\rangle = |u_n^{*\beta}(-\mathbf{k})\rangle |\alpha\rangle - |u_n^{*\alpha}(-\mathbf{k})\rangle |\beta\rangle$$

**Kramars degenerates (yellow points)**

$$|u_{2n}(\mathbf{k})\rangle = \Theta|u_{2n-1}(-\mathbf{k})\rangle$$

In this calcuation, the eigenvalue problems are solved on the half of integration interval, in other words, the quarter of Brillouin zone as shown in Fig. 75. When we perform the integrals on the other

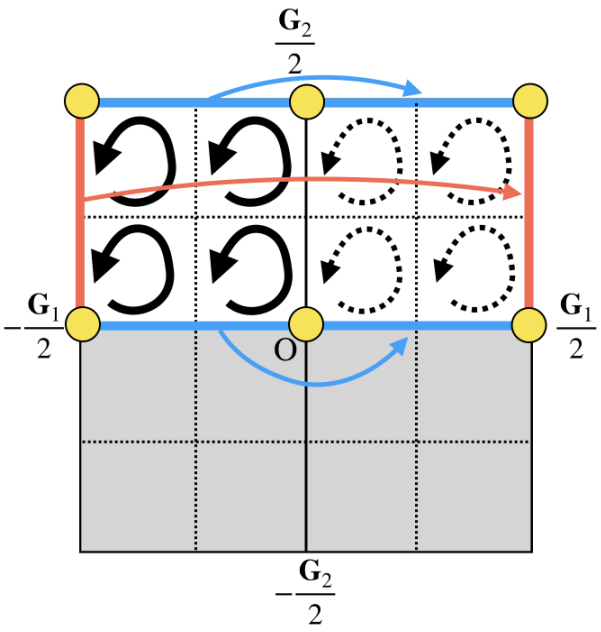



Figure 75: Gauge fixing on the half Brillouin zone. We fix the wavefunction's gauges as red parts satisfying the translational symmetry, blue parts satisfying the time-reversal symmetry, and yellow points satisfying the Kramars degeneracies.

area, we generate wave functions by fixing wavefunction's gauges on the symmetrically corresponding plaquette, and perform the integral.

In case of the three dimensional system, the Brillouin zone has six time-reversal invariant planes, $k_n = 0$, $k_n = \frac{\mathbf{G}_n}{2} (n = 1, 2, 3)$. Thus, six $Z_2$ invariants $(x_0, x_\pi, y_0, y_\pi, z_0, z_\pi)$ can be defined as shown in Fig. 76. Note that these invariants satisfy the following equation:

$$x_0 + x_\pi = y_0 + y_\pi = z_0 + z_\pi \pmod 2$$

Therefore, only four invariants become independent parameters. Based on the fact, the $Z_2$ invariant in 3D system is defined as

$$Z_2 = (\nu_0, \nu_1, \nu_2, \nu_3) = (x_0 + x_\pi, x_\pi, y_\pi, z_\pi)$$

Especially, the system of $\nu_0 = 1$ is called strong topological insulator because the $Z_2 = 1$ state appears on all the direction in the Brillouin zone.

## 56.2 Example

As an example, let us calculate a $Z_2$ invariant of a 3D topological insulator $Bi_2Se_3$ [83].

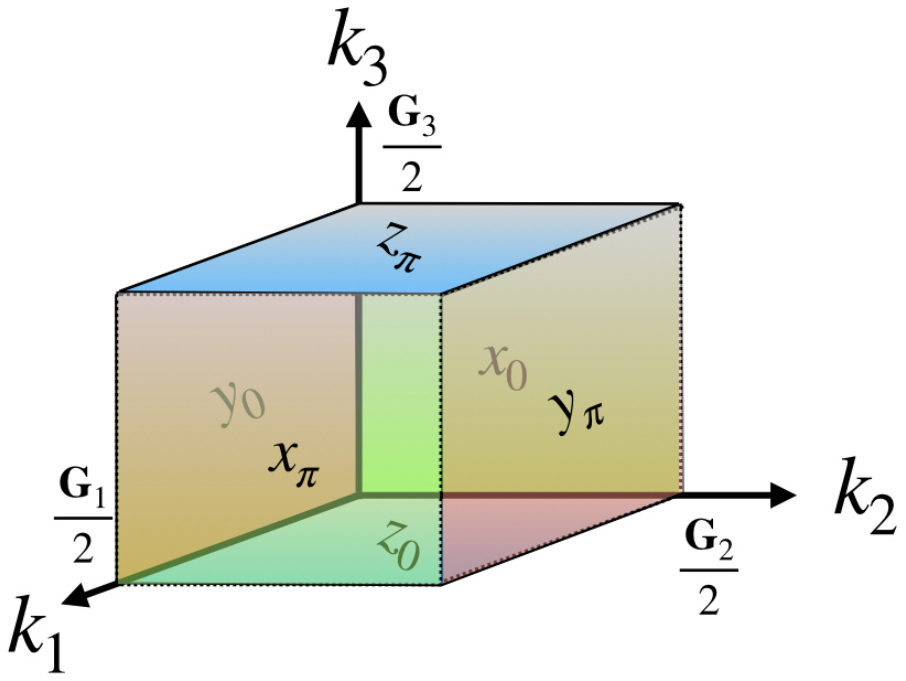



Figure 76: The six $Z_2$ invariants defined on 3D reciprocal lattice space. These invariants satisfy $x_0 + x_\pi = y_0 + y_\pi = z_0 + z_\pi$ (mod 2), and only four invariants become independent parameters.

**SCF calculation**

You can perform the SCF calculation using 'Bi2Se3-Z2.dat' stored in the directory 'work' with the following keyword:

```
HS.fileout      on      #on|off, default=off
```

After finishing the SCF calculation normally, the out file 'Bi2Se3-Z2.scfout' is generated.

**Calculation of $Z_2$ invariant**

Before computing the $Z_2$ invariant by using the code 'Z2FH.c', please compile the code in directory 'source' as

```
% make Z2FH
```

After the compilation, you may obtain the excutable file 'Z2FH' in the directory 'work'. Then, let us move on the calculation of the $Z_2$ invariant as

```
  % ./Z2FH Bi2Se3-Z2.scfout
or
  % ./Z2FH Bi2Se3-Z2.scfout < Z2FH.in > Z2FH.out
or
  % mpirun -np 4 ./Z2FH Bi2Se3-Z2.scfout < Z2FH.in > Z2FH.out
```

The file 'Z2FH.in' contains parameters which are requested by 'Z2FH' such as

```
5
1
1
```

In the first case above, you will be interactively asked from the program as follows:

```
 ****************************************************************
 ****************************************************************
  Z2FH:
  code for calculating the Z2 invariant of bulk systems
  by Fukui-Hatsugai method.
  Copyright (C), 2019, Hikaru Sawahata, Naoya Yamaguchi,
  Fumiyuki Ishii and Taisuke Ozaki
  This is free software, and you are welcome to
  redistribute it under the constitution of the GNU-GPL.

  Please cite the following article:
  H. Sawahata, N. Yamaguchi, H. Kotaka and F. Ishii,
  Jpn. J. Appl. Phys. 57, 030309 (2018).
 ****************************************************************
 ****************************************************************
 Mesh1 Number(Half Direction):5
 Calculate All plane?(0:No,1:Yes)1
 Restart?[1:x0,2:xpi,3:y0....]1

 Read the scfout file (Bi2Se3-Z2.scfout)
 ***
 The file format of the SCFOUT file:  3
 And it supports the following functions:
 - jx
 - polB
 - kSpin
 - Z2FH
 - calB
 ***
```

### Parameters for Z2FH

Let us explain the parameters in the input file 'Z2FH.in'.

- In the first line, you set the number of mesh on the half of integration interval, in other words, on the space of quarter of the first Brillouin zone (please see the Fig. 75). If you set as '5', the code performs the integrations on the square plaquettes formed by 5×5 k-point mesh in the quarter Brillouin zone.
- In the second line, you can speficy a flag whether computing $Z_2$ invariant is performed or not on all the six k-planes on which the $Z_2$ invariant is defined. If you want to compute only four

planes for $(\nu_0, \nu_1, \nu_2, \nu_3)$, you can set as '0'. Otherwise, please set as '1', corresponding to all the calculations of the six planes for $(x_0, x_\pi, y_0, y_\pi, z_0, z_\pi)$.

- In the third line, you can set the **k**-plane on which you want to restart the calculation. The numbers 1, 2, 3, 4, 5, and 6 correspond to $k_1 = 0$, $k_1 = \frac{\mathbf{G}_1}{2}$, $k_2 = 0$, $k_2 = \frac{\mathbf{G}_2}{2}$, $k_3 = 0$, and $k_3 = \frac{\mathbf{G}_3}{2}$ planes, respectively.

**Output files**

After the calculation by 'Z2FH', the following files are generated.

- Z2.dat

  This file stores the computational result of $Z_2$ invariant. In case of computing all the **k**-plane (you set the second row parameter as '1'.), the $Z_2$ invariants on the six **k**-planes are on the first line, and four of $Z_2$ invariant are on the second line as shown below:

```
Z2 invariant:(x0,xpi,y0,ypi,z0,zpi)=(1.000000,-0.000000,1.000000,-0.000000,1.000000,-0.000000)
Z2 invariant:(nu0,nu1,nu2,nu3):(1,0,0,0)
```

- LCNum*.dat

  Data files of integer-valued field $n(\mathbf{k})$. The '*' behind 'LCNum' is the index running from 1 to 6. The number (1,2,3,4,5,6) corresponds to $(x_0, x_\pi, y_0, y_\pi, z_0, z_\pi)$, respectively. When you calculate only four planes, four files 'LCNum(2,4,5,6).dat' are generated, corresponding to $(x_\pi, y_\pi, z_0, z_\pi)$.

- LCNum*.pl

  Script files for gnuplot. The '*' behind 'LCNum' is the index running from 1 to 6. The number (1,2,3,4,5,6) corresponds to $(x_0, x_\pi, y_0, y_\pi, z_0, z_\pi)$, respectively. When you calculate only four planes, four files 'LCNum(2,4,5,6).dat' are generated, corresponding to $(x_\pi, y_\pi, z_0, z_\pi)$. You can visualize of integer-valued field $n(\mathbf{k})$ by

```
% gnuplot LCNum1.pl
```

- temporal_12345.input

  This is a copy of input file, which was used for the SCF calculation, reconstructed from the scfout file.

As an example, we show the integer-valued field $n(\mathbf{k})$ on $k_1 = 0$ in Fig. 77, which can be obtained by the procedure explained above as 'gnuplot LCNum1.pl'. Since $n(\mathbf{k})$ is the gauge dependent value, you may find a different result in your calculation, while the $Z_2$ invariant should be reproduced.

## 56.3 Input files

For user's convenience, input files of four examples can be found in 'work/' as follows:

- Bi2Se3-Z2.dat

  Strong topological insulating case of Bi2Se3

- Bi2Se3_trivial-Z2.dat

  Trivial insulating case of Bi2Se3

- Bi2Se3_weak-Z2.dat

  Weak topological insulating case of Bi2Se3

- Bi111-Z2.dat

  2D topological insulating case of the Bi111 slab

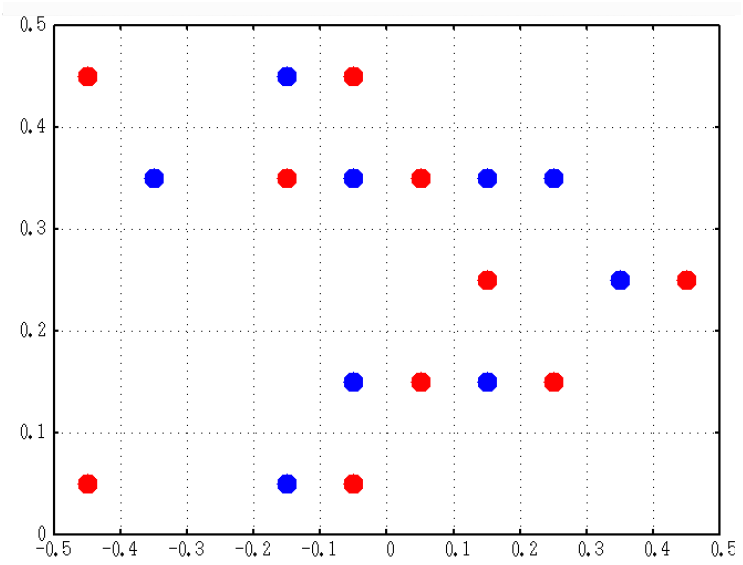



Figure 77: The example of integer-valued field $n(\mathbf{k})$. Red circle indicates +1, blue circle indicates -1, blank indicates 0. Since $n(\mathbf{k})$ is the gauge dependent value, you may find a different result in your calculation, while the $Z_2$ invariant should be reproduced.

# 57 Absolute binding energies of core levels: XPS core level energies

## 57.1 General

OpenMX supports a general method to calculate absolute binding energies of core levels in metals and insulators, based on a penalty functional and an exact Coulomb cutoff method in the framework of density functional theory [88]. With the method the spurious interaction of core holes between supercells is avoided by the exact Coulomb cutoff method, while the variational penalty functional enables us to treat multiple splittings due to chemical shift, spin-orbit coupling, and exchange interaction on equal footing. It has been demonstrated that the absolute binding energies of core levels for both metals and insulators are calculated by the proposed method in a mean absolute (relative) error of 0.4 eV (0.16%) for eight cases compared to experimental values measured with x-ray photoemission spectroscopy (XPS) within a generalized gradient approximation to the exchange-correlation functional.

By considering the energy conservation for the excitation process in the XPS measurements as schematically shown in Fig. 78, the absolute binding energies of core levels in gaseous and bulk systems are respectively given by [88]

$$E_{\rm b}^{\rm (gas)} = E_{\rm f}^{(0)}(N-1) - E_{\rm i}^{(0)}(N), \tag{12}$$

$$E_{\rm b}^{\rm (bulk)} = E_{\rm f}^{(0)}(N-1) - E_{\rm i}^{(0)}(N) + \mu_0, \tag{13}$$

where $E_{\rm i}^{(0)}(N)$ and $E_{\rm f}^{(0)}(N-1)$ are the total energies of the initial state with $(N-1)$-electrons and the final state with $N$-electrons, respectively, calculated by DFT, and $\mu_0$ is the chemical potential which is obtained from the initial state calculation. $E_{\rm f}^{(0)}(N-1)$ is evaluated by the method proposed in Ref. [88]. while $E_{\rm i}^{(0)}(N)$ is calculated by the conventional band structure calculation. Although

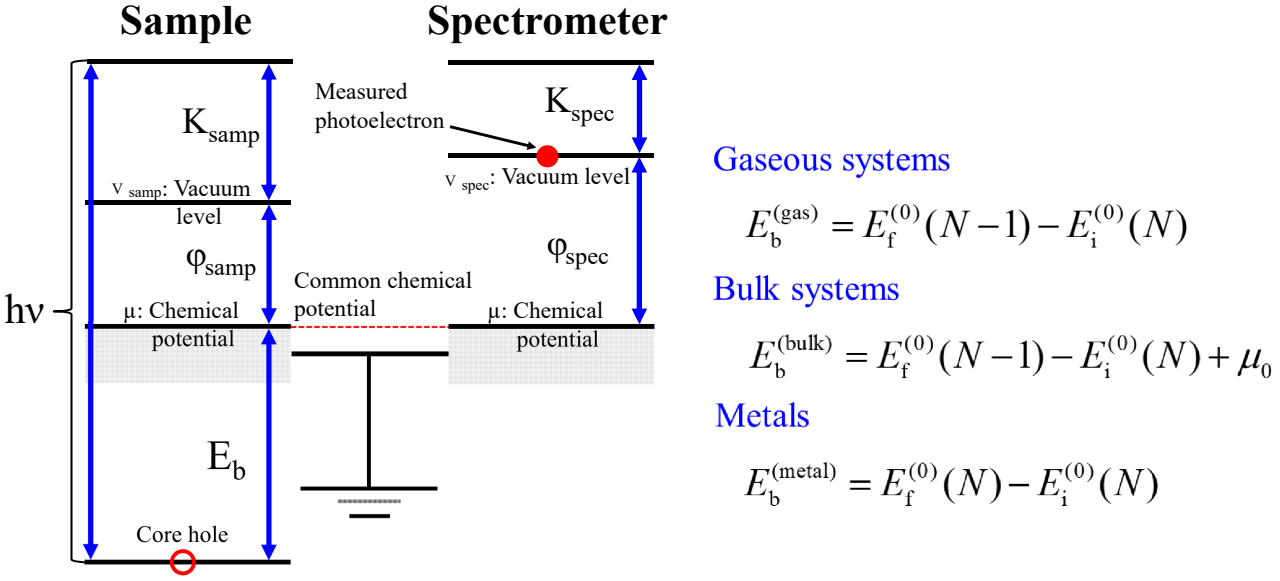



Figure 78: Schematic energy diagram for a sample and a spectrometer in the XPS measurement, and the definition of absolute binding energies of core levels in gaseous system, bulks, and metals.

Eq. (13) is applicable to both gapped and metallic systems, it should be noted that for metals Eq. (12) can be transformed using the Janak theorem [89] as

$$E_{\rm b}^{(\rm metal)} = E_{\rm f}^{(0)}(N) - E_{\rm i}^{(0)}(N), \tag{14}$$

For the final state one can calculate the total energy of the system with $N$ electrons instead of the system with $(N-1)$ electrons. The expression well matches to our intutive understanding that a perfect screeing takes place in metallic systems after the creation of core hole.

### 57.2 Gaseous systems

Let us introduce a couple of examples to illustrate how the absolute binding energies of core levels can be calculated below:

We try to calculate the binding energy for the 1s state of carbon atom in a $C_2H_2$ molecule. The **initial state** calculation can be performed as

```
% mpirun -np 4 ./openmx C2H2.dat
```

where the input file 'C2H2.dat' is avalilable in the directory 'work'. In the initial state calculation you need to specify the following keyword:

```
scf.coulomb.cutoff          on       # default = off
scf.SpinPolarization        on or nc # default = off
```

In case of 'scf.coulomb.cutoff=on', the clasical Coulomb interaction between supercells is cut off along all the three directions **a**-, **b**-, and **c**-axes using the exact Coulomb cutoff method [91]. In the calculations for the initial and final state calculations, you need to specify either 'on' or 'nc' for the keyword 'scf.SpinPolarization' and keep the same option in both the initial and final state calculations, since the system is spin-polarized after the creation of a core hole in the final state calculation. To calculate the absolute binding energies of core levels, we need to have pseudopotentials including the relevant core state. In the input file 'C2H2.dat', the following pseudopotentials are specified:

```
<Definition.of.Atomic.Species
  H   H7.0-s3p2           H_PBE19
  C   C7.0_1s-s4p3d2      C_PBE19_1s
  C1  C7.0_1s_CH-s4p3d2   C_PBE19_1s
Definition.of.Atomic.Species>
```

The pseudopotential of 'C_PBE19_1s.vps' actually includes the 1s, 2s, and 2p states as valence states. The basis sets of 'C7.0_1s.pao' and 'C7.0_1s_CH' are used for the initial and final state calculations, respectively. In the final state calculation, the radial wave functions are largely modified due to the core hole compared to the the state without the core hole. Thus, the basis set optimized for the state with the core hole has to be utilized to obtain a convergent result. The pseudopotential and basis sets for the final state calculations are available at the following website:

```
https://t-ozaki.issp.u-tokyo.ac.jp/vps_pao_core2019/
```

The data for only seven elements: B, C, N, O, Si, S, Ge, Pt are now available on the website. We have been planing to develop the pseudopotentials and basis sets of the other elements for core level excitations in the near future. The geometrical structure is specified as follows:

```
Atoms.Number         4
Atoms.SpeciesAndCoordinates.Unit   Ang # Ang|AU
<Atoms.SpeciesAndCoordinates           # Unit=Ang.
  1  C   0.6005  0.000  0.000  3.0 3.0
  2  C  -0.6005  0.000  0.000  3.0 3.0
  3  H   1.8015  0.000  0.000  0.5 0.5
  4  H  -1.8015  0.000  0.000  0.5 0.5
Atoms.SpeciesAndCoordinates>
```

It should be noted that the species for atom 1 is 'C' for which we allocate 'C7.0_1s.pao' being the basis set for the initial state, while we are going to create a core hole of 1s state for atom 1.
The **final state** calculation can be performed as

```
% mpirun -np 4 ./openmx C2H2-CH.dat
```

where the input file 'C2H2-CH.dat' is avalilable in the directory 'work'. The geometrical structure is specified as follows:

```
Atoms.Number         4
Atoms.SpeciesAndCoordinates.Unit   Ang # Ang|AU
<Atoms.SpeciesAndCoordinates           # Unit=Ang.
  1  C1  0.6005  0.000  0.000  3.0 3.0
  2  C  -0.6005  0.000  0.000  3.0 3.0
  3  H   1.8015  0.000  0.000  0.5 0.5
  4  H  -1.8015  0.000  0.000  0.5 0.5
Atoms.SpeciesAndCoordinates>
```

The atomic positions are exactly the same as for the initial state calculation, which means that the atomic relaxation during the excitation process is not taken into account. It is important to note that the species for atom 1 is 'C1' for which we allocate 'C7.0_1s_CH.pao' being the basis set for the final state. For the final state calculation, you need to specify the following keywords:

```
scf.system.charge            1.0    # default=0.0
scf.coulomb.cutoff            on    # default = off
scf.core.hole                 on    # default = off
<core.hole.state
  1 s 1
core.hole.state>
```

Considering that the final state has $(N-1)$ electrons, the number of electrons is reduced by 1 with the keyword 'scf.system.charge'. Then, we again use the Coulomb cutoff method [91] using the keyword 'scf.coulomb.cutoff' to avoid the spurious interaction of core holes between supercells. If the Coulomb cutoff method is not employed, a compensation uniform charge automatically introduced to avoid the

Table 12: Orbital index for the collinear and non-collinear cases

| Collinear case | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $s$ | 1: $s\uparrow$ | 2: $s\downarrow$ | | | | | | |
| $p$ | 1: $p_x\uparrow$ | 2: $p_y\uparrow$ | 3: $p_z\uparrow$ | 4: $p_x\downarrow$ | 5: $p_y\downarrow$ | 6: $p_z\downarrow$ | | |
| $d$ | 1: $d_{3z^2-r^2}\uparrow$ | 2: $d_{x^2-y^2}\uparrow$ | 3: $d_{xy}\uparrow$ | 4: $d_{xz}\uparrow$ | 5: $d_{yz}\uparrow$ | | | |
| | 6: $d_{3z^2-r^2}\downarrow$ | 7: $d_{x^2-y^2}\downarrow$ | 8: $d_{xy}\downarrow$ | 9: $d_{xz}\downarrow$ | 10: $d_{yz}\downarrow$ | | | |
| $f$ | 1: $f_{5z^2-3r^2}\uparrow$ | 2: $f_{5xz^2-xr^2}\uparrow$ | 3: $f_{5yz^2-yr^2}\uparrow$ | 4: $f_{zx^2-zy^2}\uparrow$ | 5: $f_{xyz}\uparrow$ | 6: $f_{x^3-3xy^2}\uparrow$ | 7: $f_{3yx^2-y^3}\uparrow$ | |
| | 8: $f_{5z^2-3r^2}\downarrow$ | 9: $f_{5xz^2-xr^2}\downarrow$ | 10: $f_{5yz^2-yr^2}\downarrow$ | 11: $f_{zx^2-zy^2}\downarrow$ | 12: $f_{xyz}\downarrow$ | 13: $f_{x^3-3xy^2}\downarrow$ | 14: $f_{3yx^2-y^3}\downarrow$ | |
| Non-collinear case | | | | | | | | |
| $s$ | 1: $J=1/2$ $M=1/2$ | 2: $J=1/2$ $M=-1/2$ | | | | | | |
| $p$ | 1: $J=3/2$ $M=3/2$ | 2: $J=3/2$ $M=1/2$ | 3: $J=3/2$ $M=-1/2$ | 4: $J=3/2$ $M=-3/2$ | 5: $J=1/2$ $M=1/2$ | 6: $J=1/2$ $M=-1/2$ | | |
| $d$ | 1: $J=5/2$ $M=5/2$ | 2: $J=5/2$ $M=3/2$ | 3: $J=5/2$ $M=1/2$ | 4: $J=5/2$ $M=-1/2$ | 5: $J=5/2$ $M=-3/2$ | 6: $J=5/2$ $M=-5/2$ | | |
| | 7: $J=3/2$ $M=3/2$ | 8: $J=3/2$ $M=1/2$ | 9: $J=3/2$ $M=-1/2$ | 10: $J=3/2$ $M=-3/2$ | | | | |
| $f$ | 1: $J=7/2$ $M=7/2$ | 2: $J=7/2$ $M=5/2$ | 3: $J=7/2$ $M=3/2$ | 4: $J=7/2$ $M=1/2$ | 5: $J=7/2$ $M=-1/2$ | 6: $J=7/2$ $M=-3/2$ | 7: $J=7/2$ $M=-5/2$ | 8: $J=7/2$ $M=-7/2$ |
| | 9: $J=5/2$ $M=5/2$ | 10: $J=5/2$ $M=3/2$ | 11: $J=5/2$ $M=1/2$ | 12: $J=5/2$ $M=-1/2$ | 13: $J=5/2$ $M=-3/2$ | 14: $J=5/2$ $M=-5/2$ | | |

Coulomb divergence in periodic charged systems. Since the treatment leads to a spurious interaction, the comparison between the initial and final states for the total energy cannot be simply performed. The core hole state that we target can be specified by the keyword 'core.hole.state'. The first number is the atomic serial index which is specified in the keyword 'Atoms.SpeciesAndCoordinates', the second symbol specifies the target $l$-channel which can be '$s$', '$p$', '$d$', or '$f$'. The last number specifies the orbital index which can be 1 to $4l+1$. The relation between the index and the state is given in Table 12. It it noted that the target state that we create a core hole is the lowest state with the $l$-channel included in the pseudopotential as valence states. In the case of $C_2H_2$ molecule, the pseudopotential of 'C_PBE19_1s.vps' includes the 1s state as valence state. Thus, the following specification:

```
<core.hole.state
  1 s 1
core.hole.state>
```

means that the state of $|1s\uparrow\rangle$ on atom 1 is chosen as target state for 'scf.SpinPolarization=on'. When a $s$-state is chosen as target state, 'scf.SpinPolarization=on' can be specified, while 'scf.SpinPolarization=nc' can also be employed. On the other hand, 'scf.SpinPolarization=nc' has to be specified to include spin-orbit coupling when a $p$-, $d$-, or $f$-state is chosen as target state.

After finishing the calculations for the initial and final states, you may obtain the total energies from the out files as

```
Initial state: -76.787732114928 (Hartree)
Final state:   -66.084858926233 (Hartree)
```

Then, using Eq. (12) the binding energy is found to be

$$\begin{aligned} E_{\rm b}^{\rm (gas)} &= E_{\rm f}^{(0)}(N-1) - E_{\rm i}^{(0)}(N), \\ &= -66.084858926233 - (-76.787732114928) = 10.702873188695({\rm Hartree}), \\ &\approx 291.24({\rm eV}). \end{aligned}$$

The obtained value of 291.24 eV is well compared to an experimental value of 291.14 eV [90]. The other examples of calculations and input files used for the calculations can be found in the website: https://t-ozaki.issp.u-tokyo.ac.jp/vps_pao_core2019/.

### 57.3 Bulk systems

Let us illustrate the calculation for the absolute binding energies of core levels in bulk by introducing TiC bulk as an example. The initial state calculation can be performed by

```
% mpirun -np 112 ./openmx TiC216.dat | tee TiC216.std
```

where any special keyword is not specified, but the spin-polarized calculation is performed with 'scf.SpinPolarization=on'. The input file 'TiC216.dat' is available in the directory 'work'. The final state calculation can be performed by

```
% mpirun -np 112 ./openmx TiC216-CH3.dat | tee TiC216-CH3.std
```

The input file 'TiC216-CH3.dat' is available in the directory 'work'. In the input file the atomic species are defined by

```
<Definition.of.Atomic.Species
  Ti  Ti7.0-s3p2d2        Ti_PBE13
  C   C6.0_1s-s3p2d1      C_PBE17_1s
  C1  C6.0_1s_CH-s3p2d1   C_PBE17_1s
Definition.of.Atomic.Species>
```

and the species of 'C1' is allocated for atom 5 as

```
Atoms.Number         216
Atoms.SpeciesAndCoordinates.Unit   Ang # Ang|AU
<Atoms.SpeciesAndCoordinates
```

```
  1  Ti   0.000000000000  0.000000000000  0.000000000000   6.0 6.0
  2  Ti   2.163500000000  2.163500000000  0.000000000000   6.0 6.0
  3  Ti   0.000000000000  2.163500000000  2.163500000000   6.0 6.0
  4  Ti   2.163500000000  0.000000000000  2.163500000000   6.0 6.0
  5  C1   2.163500000000  0.000000000000  0.000000000000   3.0 3.0
  6  C    0.000000000000  2.163500000000  0.000000000000   3.0 3.0
  7  C    0.000000000000  0.000000000000  2.163500000000   3.0 3.0
  8  C    2.163500000000  2.163500000000  2.163500000000   3.0 3.0
  ....
  ..
Atoms.SpeciesAndCoordinates>
```

Then, a core hole is created for the 1$s$-state on atom 5 by

```
scf.restart                 on
scf.restart.filename        TiC216
scf.coulomb.cutoff          on
scf.core.hole               on
scf.system.charge          0.0       # default=0.0

<core.hole.state
  5 s 1
core.hole.state>
```

The Hartree potential $V_H$ in the final state calculation consists of two contributions [88]: periodic part $V_H^{(\mathrm{P})}$ and non-periodic part $V_H^{(\mathrm{NP})}$ as

$$V_H(\mathbf{r}) = V_H^{(\mathrm{P})}(\mathbf{r}) + V_H^{(\mathrm{NP})}(\mathbf{r}), \tag{15}$$

and the periodic part $V_H^{(\mathrm{P})}$ is calculated by the charge density obtained in the initial state calculation via the Poisson equation with the periodic boundary condition. One can specify the charge density obtained in the initial state calculation by keywords 'scf.restart' and 'scf.restart.filename'. The non-periodic part $V_H^{(\mathrm{NP})}$ is calculated by an exact Coulomb cutoff method [91] with the difference charge density $\Delta\rho(\mathbf{r}) = \rho_{\mathrm{f}}(\mathbf{r}) - \rho_{\mathrm{i}}(\mathbf{r})$, where $\rho_{\mathrm{f}}(\mathbf{r})$ and $\rho_{\mathrm{i}}(\mathbf{r})$ are charge densities of final and initial states, respectively, and the cutoff radius for the Coulomb interaction is set to the half of the lenght of the shortest lattice vector. You need to switch on the keyword 'scf.coulomb.cutoff' to enable the exact Coulomb cutoff method. The core hole is created by the keywords 'scf.core.hole' and 'core.hole.state'. In this case, a core hole for the 1s state on atom 5 is created. It should be noted that the keyword 'scf.system.charge' is set to 0.0, since the TiC bulk is a metal. When you treat a gapped system, 'scf.system.charge' has to be set to 1.0.

After finishing the calculations for the initial and final states, you may obtain the total energies from the out files as

```
Initial state: -10499.900104007471 (Hartree)
Final state:   -10489.553360141708 (Hartree)
```

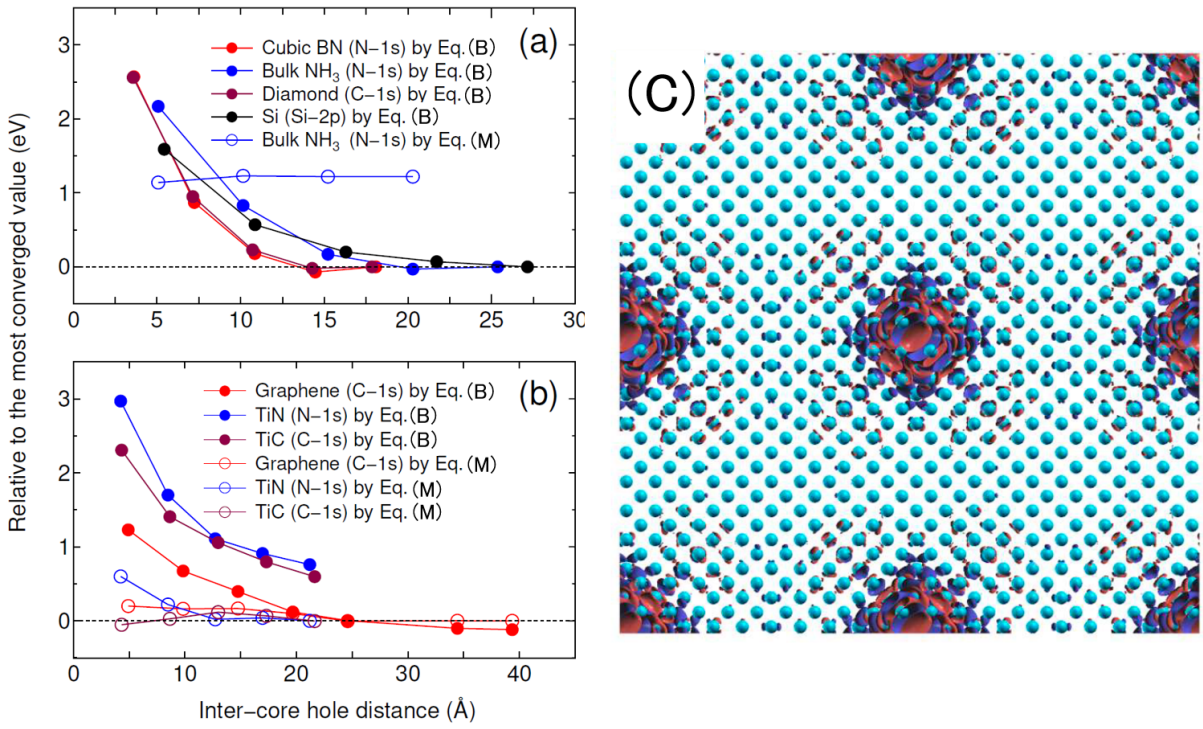



Figure 79: Calculated binding energies, relative to the most converged value, of (a) gapped systems and (b) a semimetal (graphene) and metals as a function of inter-core hole distance. The reference binding energies in (a) and (b) were calculated by Eqs. (13) and (14), respectively, for the largest unit cell for each system. In Figs. Eqs. (B) and (M) correspond to Eqs. (13) and (14), respectively. (c) Difference charge density in silicon, induced by the creation of a core hole in the 2p states, where the unit cell contains 1000 atoms and the intercore hole distance is 27.15 Å.

Then, using Eq. (14) the binding energy is found to be

$$\begin{aligned} E_{\rm b}^{\rm (metal)} &= E_{\rm f}^{(0)}(N-1) - E_{\rm i}^{(0)}(N), \\ &= -10489.553360141708 - (-10499.900104007471) = 10.346743865763({\rm Hartree}), \\ &\approx 281.55({\rm eV}). \end{aligned}$$

The obtained value of 281.55 eV is well compared to an experimental value of 281.5 eV [92].

As an example of gapped systems, let us introduce calculations for bulk silicon. One can perform the initial and final state calculations as

```
% mpirun -np 256 ./openmx Si-4-SOI.dat | tee Si-4-SOI.std
% mpirun -np 256 ./openmx Si-4-CH-SOI1.dat | tee Si-4-CH-SOI1.std
% mpirun -np 256 ./openmx Si-4-CH-SOI6.dat | tee Si-4-CH-SOI6.std
```

The input file of 'Si-4-SOI.dat' is for the initial state calculation, while 'Si-4-CH-SOI1.dat' and 'Si-4-CH-SOI6.dat' are for the final state calculations with a core hole for the $2p$ state on Si atom specified by $2p_{3/2}(J = 3/2, M = 3/2)$ and $2p_{1/2}(J = 1/2, M = -1/2)$, respectively. To take account of the spin-orbit interaction in the $2p$ state on the Si atom the non-collinear calculations are performed by specifying the following keywords:

```
scf.SpinPolarization        nc          # On|Off|NC
scf.SpinOrbit.Coupling      on          # On|Off, default=off
```

After finishing the calculations for the initial and final states, you may obtain the total energies from the out files as

```
Initial state:          -34820.483255130872 (Hartree)
Final state for SOI1:   -34816.628201335407 (Hartree)
Final state for SOI6:   -34816.601864921540 (Hartree)
```

and the chemical potential can be obtained from the initial state calculation. Then, using Eq. (13) the binding energies for SOI1 and SOI6 are found to be

For $2p_{3/2}$

$$\begin{aligned} E_{\rm b}^{\rm (bulk)} &= E_{\rm f}^{(0)}(N-1) - E_{\rm i}^{(0)}(N-1) + \mu_0, \\ &= -34816.628201335 - (-34820.483255131) - 0.201641583 = 3.65341221({\rm Hartree}), \\ &\approx 99.41({\rm eV}), \end{aligned}$$

For $2p_{1/2}$

$$\begin{aligned} E_{\rm b}^{\rm (bulk)} &= E_{\rm f}^{(0)}(N-1) - E_{\rm i}^{(0)}(N-1) + \mu_0, \\ &= -34816.601864922 - (-34820.483255131) - 0.201641583 = 3.6797486({\rm Hartree}), \\ &\approx 100.13({\rm eV}). \end{aligned}$$

The obtained values of 99.41 and 100.13 eV are well compared to an experimental value of 99.2 and 99.8 eV for $2p_{3/2}$ and $2p_{1/2}$, respectively [92], where the degeneracies of $2p_{3/2}$ and $2p_{1/2}$ are 4 and 2, repectively, as can be seen from Table 12. It should be noted that for gapped systems the convergence of the absolute binding energies of core levels is slow with repect to the cell size. In Figs. 79(a) and (b), the relative binding energies are shown as a function of inter-core hole distance for gapped systems and metals, respectively. It is found that the convergence is slow for the gapped systems, implying that a large supercell needs to be used to obtain the convergence result. On the other hand, we see that for metals the binding energies quickly converges at a relatively small inter-core hole distance. In Fig. 79(c) we show difference charge density in silicon bulk, induced by the creation of a core hole in the $2p$ state. To screen the potential produced by the core hole, a charge redistribution takes place up to around $\sim 7$ Å from the core hole. The slow convergence found in gapped systems is attributed to the charge redistribution. On the other hand, in metals the charge redistribution is relatively short range, resulting in the fast convergence as shown in Fig. 79(b).

The other examples of calculations and input files used for the calculations can be found in the website: https://t-ozaki.issp.u-tokyo.ac.jp/vps_pao_core2019/. Also, applications of the method for silicene, borophene, and single Pt atoms dispersed on graphene can be found in Refs. [93, 94, 95].

### 57.4 Examples

For user's convenience, input files for three examples can be found in the directory 'work' as follows:

- C2H2.dat, C2H2-CH.dat

  Calculation of the absolute binding energy of 1s state on C atom in a gaseous $C_2H_2$ molecule.

- TiC216.dat, TiC216-CH3.dat

  Calculation of the absolute binding energy of 1s state on C atom in metallic TiC216 bulk.

- Si-4-SOI.dat, Si-4-CH-SOI1.dat, Si-4-CH-SOI6.dat

  Calculation of the absolute binding energies of $2p_{3/2}$ and $2p_{1/2}$ state on Si atom in Si bulk.

The other examples of calculations and input files used for the calculations can be found in the website: https://t-ozaki.issp.u-tokyo.ac.jp/vps_pao_core2019/.

# 58 Ionization potential and electron affinity of gaseous systems

The ionization potential and electron affinity of gaseous systems can be calculated by a delta SCF method with an exact Coulomb cutoff method [91]. With the exact Coulomb cutoff method, a calculation for a charged isolated system is possible even in the periodic boundary condition. What you need is to perform two calculations for the ground and ionized states of an isolated system, and to calculate the difference of the total energies between them. Let us illustrate calculations for ionization potential of gaseous systems. The first example is a water molecule. One can perform a calculation for the ground state as

```
% mpirun -np 3 ./openmx H2O+0.dat | tee h2o+0.std
```

The input file 'H2O+0.dat' is available in the directory 'work'. The geometry structure was optimized with the same computational condition before the calculation. To avoid the Coulomb interaction between the supercells the exact Coulomb cutoff method [91] is employed by the following keyword:

```
scf.coulomb.cutoff            on     # default=off, on|off
```

Even if the method is employed, where the cutoff radius for the Coulomb interaction is set to the half of the lenght of the shortest lattice vector, the cell size has to be large enough so that the Coulomb interaction in the central cell can be properly calculated.

The calculation for the ionized state can be performed as

```
% mpirun -np 3 ./openmx H2O+1.dat | tee h2o+1.std
```

The input file 'H2O+1.dat' is available in the directory 'work'. Compared to 'H2O+0.dat' the following keywords need to be changed:

```
scf.system.charge            1.0       # default=0.0
scf.coulomb.cutoff            on       # on|off, default=off
scf.SpinPolarization          on       # On|Off|NC
```

The system is positively charged up by the keyword 'scf.system.charge'. The Coulomb divergence in the charged systems can be eliminated by using the exact Coulomb cutoff method with 'scf.coulomb.cutoff'. The system may be spin-polarized after the ionization. Thus, 'scf.SpinPolarization' is switched on. After finishing the two caculations you may obtain the total energies from the out files as

```
Ground state:          -17.477268421216 (Hartree)
Charged state of +1:   -17.010776518028 (Hartree)
```

Then, the ionization potential IP, defined to be (total energy of charged state of +1) - (total energy of the ground state), is calculated as

$$\begin{aligned} \mathrm{IP} &= -17.010776518028 - (-17.477268421216) = 0.466491903188(\mathrm{Hartree}), \\ &\approx 12.69(\mathrm{eV}). \end{aligned}$$

The obtained value of 12.69 eV is well compared to an experimental value of 12.65 eV [96]. As well as the ionization potential, one can calculate the electron affinity, defined to be (total energy of the ground state) - (total energy of charged state of -1), of gaseous systems by specifying

Table 13: Calculated ionization potential and electron affinity of gaseous systems. All the geometrical structures were optimized with the same computational condition before the calculations. All the input files, which are listed below, used the calculations are available in the directory 'work'.

| Ionization potential | | | |
|---|---|---|---|
| System | Expt. (eV) | Calc. (eV) | Input files |
| $H_2O$ | 12.65 [96] | 12.69 | H2O+0.dat, H2O+1.dat |
| $C_2H_2$ | 11.43 [97] | 11.47 | C2H2+0.dat, C2H2+1.dat |
| $C_2H_2$ | 10.55 [97] | 10.57 | C2H4+0.dat, C2H4+1.dat |
| $O_2$ | 12.04 [97] | 12.85 | O2+0.dat, O2+1.dat |
| CO | 14.01 [97] | 13.85 | CO+0.dat, CO+1.dat |

| Electron affinity | | | |
|---|---|---|---|
| System | Expt. (eV) | Calc. (eV) | Input files |
| OH | 1.81 [97] | 1.82 | OH-0.dat, OH-1.dat |
| $O_2$ | 0.41 [97] | -0.29 | O2-0.dat, O2-1.dat |
| $Cl_2$ | 2.37 [97] | 0.96 | Cl2-0.dat, Cl2-1.dat |
| CN | 3.88 [97] | 3.51 | CN-0.dat, CN-1.dat |
| SiH | 1.27 [97] | 1.17 | SiH-0.dat, SiH-1.dat |

```
scf.system.charge           -1.0       # default=0.0
```

With 'scf.system.charge=-1.0', the system is negatively charged up by one additional electron.

The results for benchmark calculations of the ionization potential and electron affinity of gaseous systems are shown in Table 13. We see that the calculated results of ionization potential are well compared to experimental data, while the calculated electron affinities of some systems seem to deviate from the experimental values especially for $O_2$ and $Cl_2$.

# 59 Optical conductivity and dielectric function

## 59.1 General

In OpenMX Ver. 3.9, the conductivity and dielectric function can be calculated based on the Kubo-Greenwood formula [98]. Starting from the Born approximation, the complex tensor of conductivity and dielectric function which are frequency dependent are calculated within a linear response to a perturbing frequency dependent electric field. Other physical quantities such as absorption, extinction, transmission, reflection, and refractive index are also calculated, which are all derived from the conductivity. The functionality is compatible with only the collinear calculations. The extension of the functionality to the non-collinear case will be supported in the future release. Since the multi-level parallelization has been implemented, it is possible to perform large-scale calculations of systems including more than 1000 atoms on massively parallel computers.

## 59.2 Si case

Let us illustrate the calculation using an input file 'Si2_k50x50x50.dat' stored in the directory 'work'. The input file is for a calculation of Si bulk including 2 atoms in the unit cell, where $10 \times 10 \times 10$ k-points and $50 \times 50 \times 50$ k-points are used for the SCF calculation and the calculation of conductivity, respectively. One can perform the calculation as

```
% mpirun -np 112 ./openmx Si2_k50x50x50.dat
```

After finishing the SCF calculation normally, the relevant calculation '<Optical calculation start>' starts as shown in the standard output below:

```
****************** MD= 1  SCF=17 ******************
<Poisson>  Poisson's equation using FFT...
<Set_Hamiltonian>  Hamiltonian matrix for VNA+dVH+Vxc...
<Band>  Solving the eigenvalue problem...
 KGrids1:  -0.45000  -0.35000  -0.25000  -0.15000  -0.05000   0.05000   0.15000 ....
 KGrids2:  -0.45000  -0.35000  -0.25000  -0.15000  -0.05000   0.05000   0.15000 ....
 KGrids3:  -0.45000  -0.35000  -0.25000  -0.15000  -0.05000   0.05000   0.15000 ....
<Band_DFT>  Eigen, time=0.028573
<Band_DFT>  DM, time=0.024604
    1   Si  MulP   2.0000  2.0000 sum   4.0000
    2   Si  MulP   2.0000  2.0000 sum   4.0000
 Sum of MulP: up   =     4.00000 down          =     4.00000
              total=     8.00000 ideal(neutral)=     8.00000
<DFT>  Total Spin Moment (muB) =  0.000000000000
<DFT>  Mixing_weight= 0.020000000000
<DFT>  Uele   =   -2.418066179485  dUele     =   0.000000000118
<DFT>  NormRD =    0.000000000011  Criterion =   0.000000001000

<Optical calculation start>
 CDDF.KGrids1:  -0.49000  -0.47000  -0.45000  -0.43000  -0.41000  -0.39000  -0.37000 ....
 CDDF.KGrids2:  -0.49000  -0.47000  -0.45000  -0.43000  -0.41000  -0.39000  -0.37000 ....
 CDDF.KGrids3:  -0.49000  -0.47000  -0.45000  -0.43000  -0.41000  -0.39000  -0.37000 ....
<Optical calculations end, time=24.31524 (s)>
<MD= 1>  Force calculation
  Force calculation #1
  Force calculation #2
```

```
  Force calculation #3
  Force calculation #4
  Force calculation #5
<MD= 1>  Total Energy
  Force calculation #6
  ....
  ...
```

In this case, it is found from the standard output that the computational time of the relevant calculation is about 24 second. After all the calculations finish, you obtain the following output files relevant to the functionality:

| | |
|---|---|
| Si2_k50x50x50.cd_re | real part of optical conductivity tensor |
| Si2_k50x50x50.cd_im | imaginary part of optical conductivity tensor |
| Si2_k50x50x50.df_re | real part of dielectric function tensor |
| Si2_k50x50x50.df_im | imaginary part of dielectric function tensor |
| Si2_k50x50x50.absorption | absorption tensor |
| Si2_k50x50x50.extinction | extinction tensor |
| Si2_k50x50x50.transmission | transmission tensor |
| Si2_k50x50x50.reflection | reflection tensor |
| Si2_k50x50x50.refractive_index | refractive index tensor |

The format of each file can be found in the header part of the file, where the unit for each physical quantity is also provided. For example, 'Si2_k50x50x50.cd_re' storing the real part of optical conductivity tensor $\sigma$ can be seen as shown below:

```
# conductivity tensor (real part) , unit = Siemens/meter = Mho/meter = 1/(Ohm*meter)
# index: energy-grid=1, xx=2, xy=3, xz=4, yx=5, yy=6, yz=7, zx=8, zy=9, zz=10, trace=11
#energy-grid(eV)     xx           xy           xz           yx           yy           yz           zx           zy           zz   (xx+yy+zz)/3
  0.00000  16877.3220211  -227.5621843  -227.5697597  -227.5625069  16877.3919038  -227.5042335  -227.5702078  -227.5041190  16877.3911375  16877.3683541
  0.00100  16877.3325817  -227.5625199  -227.5700960  -227.5628426  16877.4024628  -227.5045682  -227.5705442  -227.5044537  16877.4016971  16877.3789139
  0.00200  16877.3431423  -227.5628556  -227.5704323  -227.5631782  16877.4130218  -227.5049028  -227.5708805  -227.5047883  16877.4122567  16877.3894736
  0.00300  16877.3602570  -227.5634100  -227.5709878  -227.5637327  16877.4301338  -227.5054554  -227.5714360  -227.5053409  16877.4293698  16877.4065869
  0.00400  16877.3839257  -227.5641832  -227.5717624  -227.5645058  16877.4537989  -227.5062261  -227.5722106  -227.5061116  16877.4530363  16877.4302536
  ......
  ...
```

The first column is the photon energy (eV), and from the second columns onward, the components of tensor such as $\sigma_{xx}$ and $\sigma_{xy}$ are stored. In the last column the average value of the diagonal components $\sigma_{xx}$, $\sigma_{yy}$, and $\sigma_{zz}$ is given. The other output files also follow the same format as in 'Si2_k50x50x50.cd_re'. By plotting the first column as horizontal axis and the second column as vertical axis of 'Si2_k50x50x50.cd_re', 'Si2_k50x50x50.cd_im', 'Si2_k50x50x50.df_re', and 'Si2_k50x50x50.df_im', one can obtain the optical conductivity $\sigma_{xx}$ and dielectric function $\varepsilon_{xx}$ as shown in Fig. 80.

## 59.3 Relevant keywords

The specification of each keyword relevant to calculations of conductivity tensor and dielectric function is given below.

**CDDF.start**
Switch on the calculations of conductivity tensor and dielectric function when you want to perform them. The default is 'off'. In case of 'off', the calculation of conductivity tensor and dielectric function is turned off.

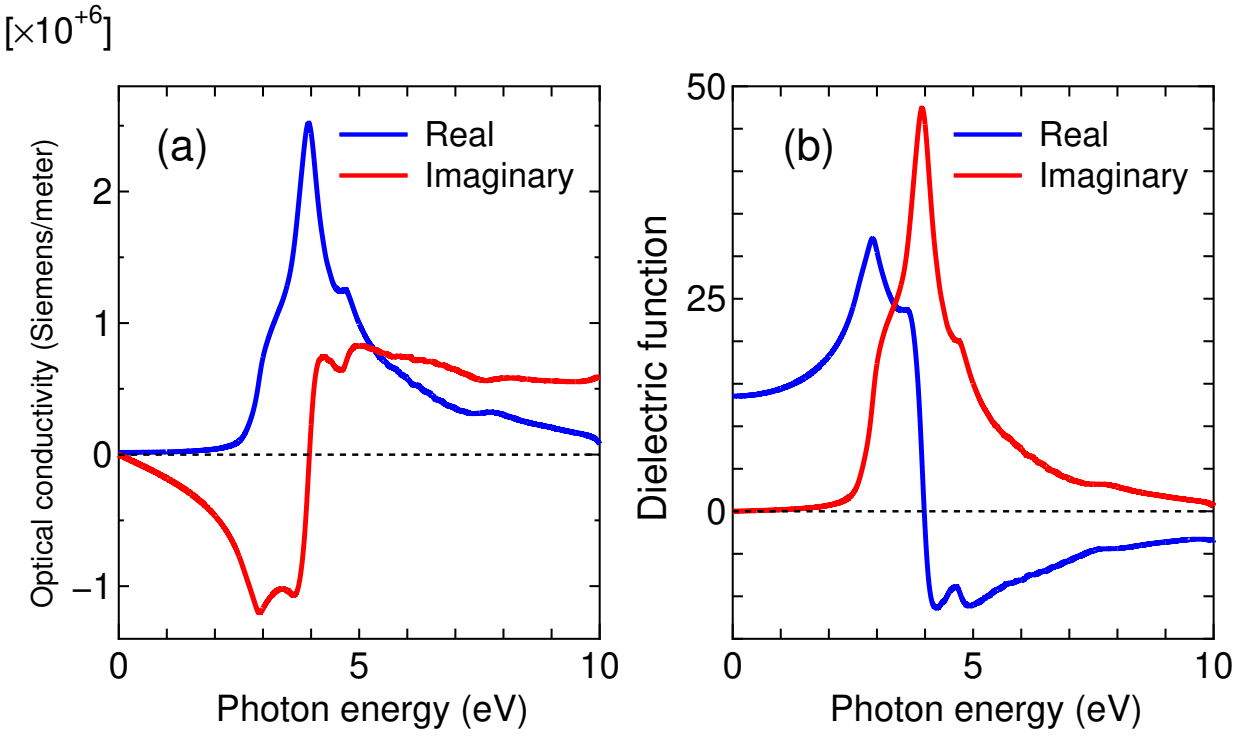



Figure 80: (a) The optical conductivity $\sigma_{xx}$ and (b) dielectric function $\varepsilon_{xx}$ of Si bulk. The second column as vertical axis of 'Si2_k50x50x50.cd_re', 'Si2_k50x50x50.cd_im', 'Si2_k50x50x50.df_re', and 'Si2_k50x50x50.df_im' are plotted as a function of the first column in each file. The input file 'Si2_k50x50x50.dat' used for the calculation is available in the directory 'work'.

```
CDDF.start    on          # on|off, default=off
```

**CDDF.FWHM**

Setting the full width at half maximum of conductivity and dielectric function. The default is '0.2' in eV. The Lorentian function is smeared out with the parameter.

```
CDDF.FWHM     0.2         # default = 0.2 (eV)
```

**CDDF.maximum_energy**

Setting the maximum energy of optical spectra for calculations of conductivity and dielectric function. The default is '10.0' in eV. The energy range begins from 0.0 eV.

```
CDDF.maximum_energy     10.0     # default = 10.0 (eV)
```

**CDDF.additional_maximum_energy**

Setting the additional energy range in the frequency domain. Although the energy range for the output of conductivity and dielectric function is specified by 'CDDF.maximum_energy', for the calculations the states beyond the energy range of the output are also taken into account, since the states beyond the energy range of the output may contribute because of the broadening of Lorentzian function. The energy range for the calculations can be controlled by a keyword 'CDDF.additional_maximum_energy'. For example, when the maximum energy is 10 eV, which is specified by 'CDDF.maximum_energy',

Table 14: Computational time of conductivity and dielectric function of Si crystal.

| # of Si atoms | Supercell | Diagonalization | k-Grid | Total time (s) (CPUs=128) | Total time (s) (CPUs=256) | Total time (s) (CPUs=512) | Total time (s) (CPUs=1024) | Total time (s) (CPUs=2048) |
|---|---|---|---|---|---|---|---|---|
| 512 atoms | 4x4x4 | Cluster | 1x1x1 | 3367.16826 | 1755.60797 | 919.21912 | 464.27761 | 253.32210 |
| | 4x4x4 | ScaLAPACK | 2x1x1 | 6819.30193 | 3499.51872 | 1838.64406 | 948.87978 | 513.79250 |
| | 4x4x4 | Band | 2x2x2 | 15300.58350 | 10217.17765 | 5953.19907 | 3518.84650 | 1747.20171 |
| 1000 atoms | 5x5x5 | Cluster | 1x1x1 | | | 6900.35370 | 3511.85143 | 1778.33693 |
| | 5x5x5 | ScaLAPACK | 2x1x1 | | | 12994.17818 | 6817.43990 | 3460.76787 |
| | 5x5x5 | Band | 2x2x2 | | | 43676.20392 | 26055.12739 | 13318.14587 |

and the additional energy range is set to 1.0 eV by 'CDDF.additional_maximum_energy', then the total energy range becomes 11.0 (10.0+1.0) eV. The default value is 0.0 eV.

```
CDDF.additional_maximum_energy         1.0     # default = 0.0 eV
```

**CDDF.frequency.grid.total_number**

Setting the total number of grids for conductivity and dielectric function. The default number of grids is '10000'. And, the interval in the energy grid is given by ( Maximum energy - 0.0 ) / total number of energy-grid, e.g. $(10.0 - 0.0)/10000 = 0.0010$ (eV).

```
CDDF.frequency.grid.total_number    10000     # default = 10000
```

**CDDF.Kgrid**

Setting a set of numbers (n1,n2,n3) of grids to discretize the first Brillouin zone in the k-space, which is used for the calculations of conductivity and dielectric function. For the reciprocal vectors $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$, and $\tilde{\mathbf{c}}$ in the k-space, please provide a set of numbers (n1,n2,n3) of grids as 'n1 n2 n3'. According to the (n1,n2,n3), a regular mesh in the first Brillouin zone will be generated. It does not need to be the same as scf.Kgrid which is used for the SCF calculation. So, one may use a rather coarse grid for the SCF calculation, and change to a finer grid for the calculations of conductivity and dielectric function to reduce the computational cost.

**CDDF.material_type**

Setting the type of material: metal or insulator. 0 is for insulator, and 1 is for insulator and metal.

```
CDDF.material_type 0     # Default=0
```

## 59.4 Benchmark calculations

A couple of examples as benchmark calculations are shown below:

**Si**

The real part of dielectric function of Si bulk is shown for a series of k-grids in Fig. 81. We see that as increasing k-grid from $10 \times 10 \times 10$ to $100 \times 100 \times 100$, the real part of dielectric function is getting converged. It is found that we need to have a fine grid for the k-points to obtain a well converged result. In Tables 14 and 15, we show the computational time and parallel efficiency in the calculation of the conductivity and dielectric function for supercells of Si bulk. The results suggest that it might be possible to treat systems including 1000 atoms if 1000 CPU cores are available.

**$\beta$-PVDF**

The real part of dielectric function of $\beta$-PVDF (polyvinylidene fluoride) is shown for a series of k-grids in Fig. 82. We see that the k-grid of $6 \times 9 \times 21$ is required to get the convergent result. In Tables 16 and

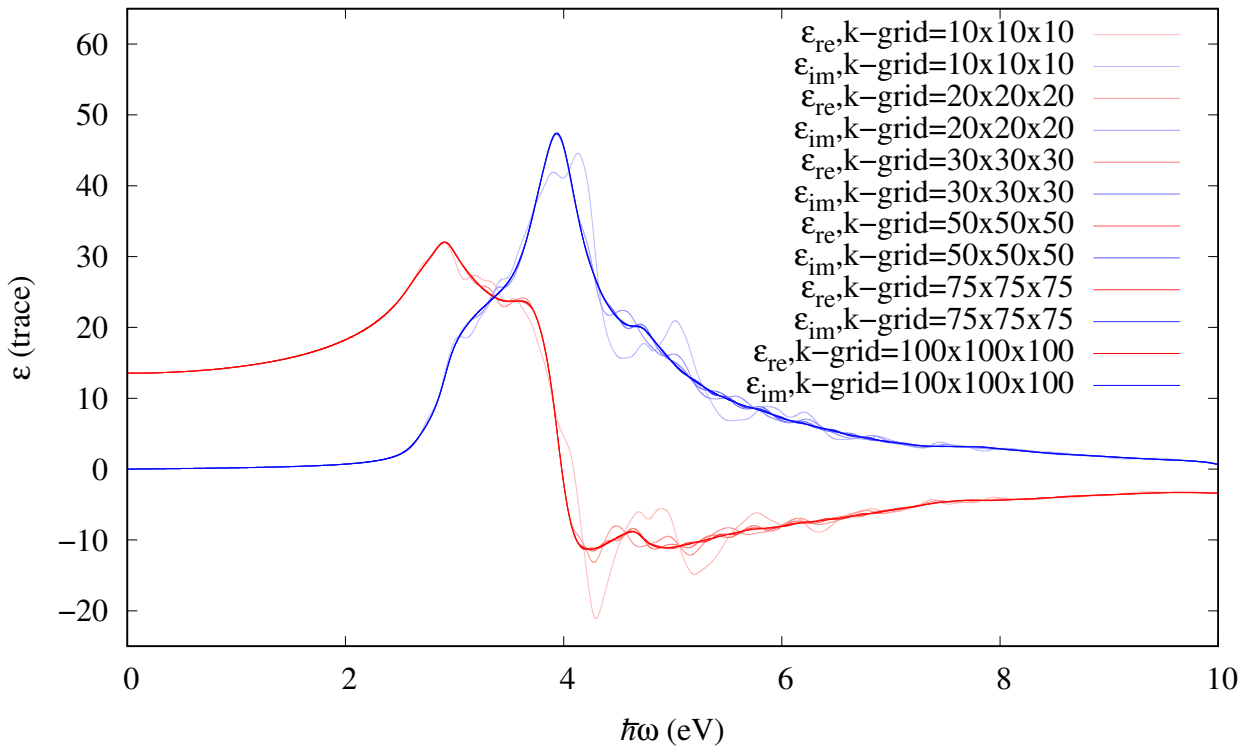



Figure 81: The dielectric function of Si crystal (primitive cell, 2 atoms) with 6 different K-grids: $10 \times 10 \times 10$, $20 \times 20 \times 20$, $30 \times 30 \times 30$, $50 \times 50 \times 50$, $75 \times 75 \times 75$, and $100 \times 100 \times 100$. The blue and red lines are real and imaginary parts of dielectric function, respectively, where CDDF.FWHM=0.2 eV was used.

17, we show the computational time and parallel efficiency in the calculation of the conductivity and dielectric function for supercells of $\beta$-PVDF. It is confirmed that the parallel efficiency is reasonably good, and the elapsed time is less than one hour when the CPU cores of 256 are used. In Fig. 82, we show the $xx$, $yy$, and $zz$ components of real part of dielectric function of $\beta$-PVDF for your reference.

**$VO_2$ in the R phase**

The real and imaginary parts of dielectric function of $VO_2$ in the R phase are shown for a series of k-grids in Fig. 84. We see that the k-grid of $16 \times 16 \times 16$ is required to get the convergent result. Tables 18 and 19 show the computational time and parallel efficiency in the calculation of the conductivity and dielectric function for supercells of $VO_2$ in the R phase. It is confirmed that the parallel efficiency is reasonably good, allowing us to treat large-scale systems in an elapsed time of 1 hour.

## 59.5 Codes

Routines relevant to the calculations of conductivity and dielectric function are listed below:

```
Band_DFT_Col_Optical.c          Band_DFT_Col_Optical_ScaLAPACK.c
Band_DFT_NonCol_Optical.c       Calc_optical.c
```

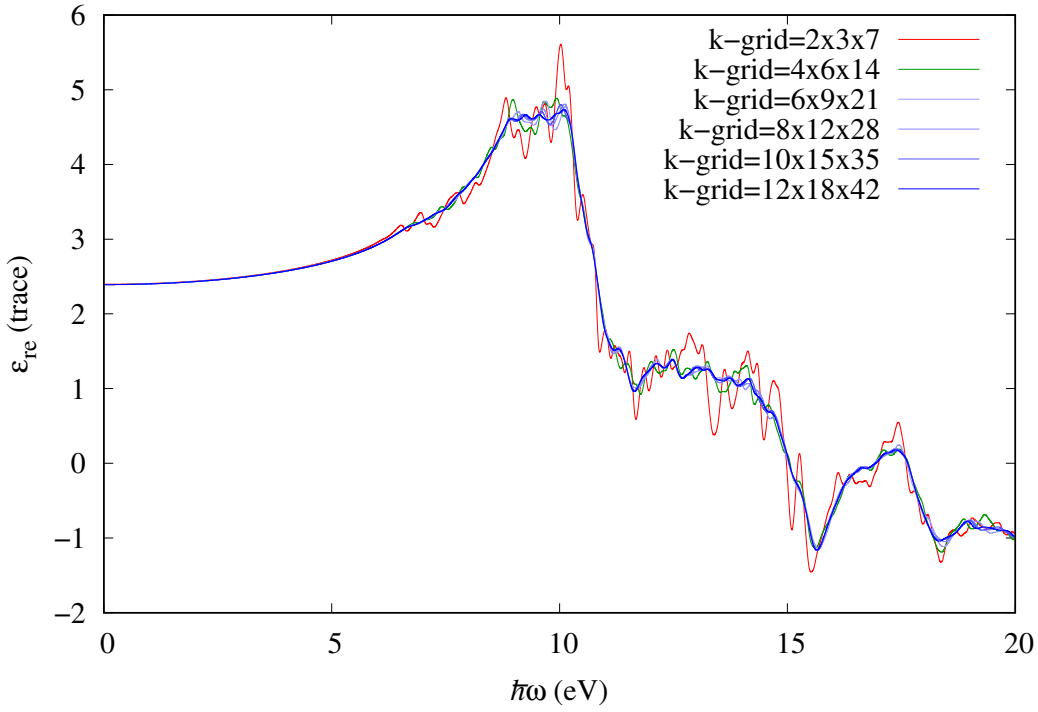



Figure 82: The real part of dielectric function of $\beta$-PVDF (polyvinylidene fluoride) consisting of 6 atoms in the $1 \times 1 \times 1$ cell for a series of k-grids. CDDF.FWHM=0.2 eV was used.

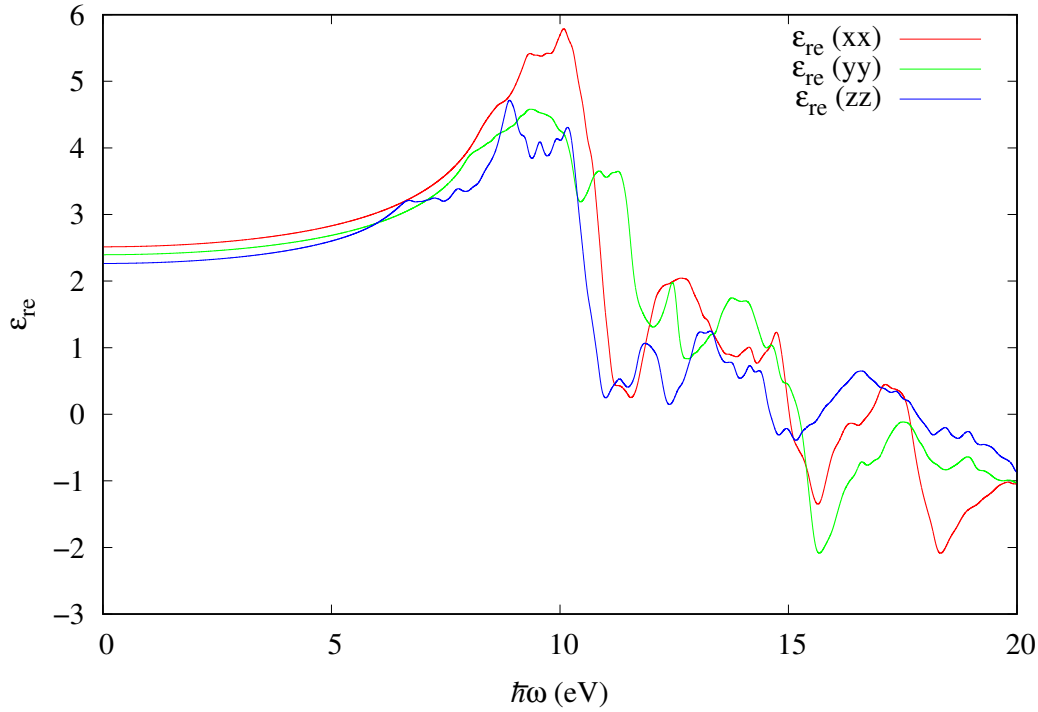



Figure 83: The $xx$, $yy$, and $zz$ components of real parts of dielectric function of $\beta$-PVDF (6 atoms, $1 \times 1 \times 1$ cell).

Table 15: Calculation time of conductivity and dielectric function of Si crystal (512 atoms, $4 \times 4 \times 4$ supercell, k-grid=$1 \times 1 \times 1$). The speed-up ratio with repect to the case with 128 CPU cores is shown in the last column.

| # of CPUs | Total time of calculating conductivity and dielectric function (s) | Ratio of total time by # of CPUs to total time by 128 CPUs |
|---|---|---|
| 128 | 3367.16826 | 1.000 |
| 256 | 1755.60797 | 1.918 |
| 512 | 919.21912 | 3.663 |
| 1024 | 464.27761 | 7.252 |
| 2048 | 253.32210 | 13.292 |

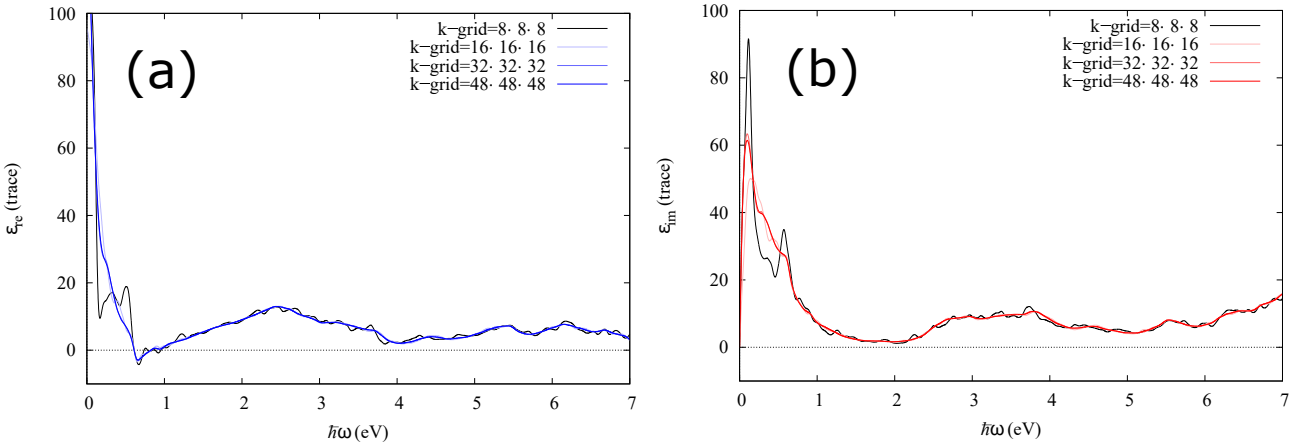



Figure 84: The real and imaginary parts of dielectric function of $VO_2$ (R phase, 6 atoms, $1 \times 1 \times 1$ cell) with a series of k-grids.

| | |
|---|---|
| `Cluster_DFT_Optical.c` | `Cluster_DFT_Optical_ScaLAPACK.c` |
| `DFT.c` | `Get_Cnt_dOrbitals.c` |
| `Get_dOrbitals.c` | `Input_std.c` |
| `NabraMatrixElements.c` | `Set_dOrbitals_Grid.c` |

## 59.6 Examples

For user's convenience, input files for five examples can be found in the directory 'work/cddf_examples' as follows:

Table 16: Total time of calculating conductivity and dielectric function of $\beta$-PVDF (polyvinylidene fluoride) consisting of 540 atoms which corresponds to the $3 \times 3 \times 5$ supercell.

| Diagonalization | k-Grid | Total time (s) (CPUs=128) | Total time (s) (CPUs=256) | Total time (s) (CPUs=512) | Total time (s) (CPUs=1024) | Total time (s) (CPUs=2048) |
|---|---|---|---|---|---|---|
| Cluster | 1x1x1 | 4215.67238 | 2086.76841 | 1061.06835 | 565.92209 | 329.42010 |
| ScaLAPACK | 1x1x2 | 3330.66855 | 1711.98413 | 877.61705 | 599.73266 | 330.11080 |
| Band | 2x2x2 | 12854.63816 | 7244.26768 | 3591.33618 | 1866.03405 | 998.07756 |

Table 17: Calculation time of conductivity and dielectric function of $\beta$-PVDF (polyvinylidene fluoride) consisting of 540 atoms which corresponds to the $3 \times 3 \times 5$ supercell, where k-grid of $1 \times 1 \times 1$ was used. The speed-up ratio with repect to the case with 128 CPU cores is shown in the last column.

| # of CPUs | Total time of calculating conductivity and dielectric function (s) | Ratio of total time by # of CPUs to total time by 128 CPUs |
|---|---|---|
| 128 | 4215.67238 | 1.000 |
| 256 | 2086.76841 | 2.020 |
| 512 | 1061.06835 | 3.973 |
| 1024 | 565.92209 | 7.449 |
| 2048 | 329.42010 | 12.797 |

Table 18: Calculation time of conductivity and dielectric function of $VO_2$ (R phase, 384 atoms, $4\times4\times4$ supercell)

| Diagonalization | k-Grid | Total time (s) (CPUs=128) | Total time (s) (CPUs=256) | Total time (s) (CPUs=512) | Total time (s) (CPUs=1024) | Total time (s) (CPUs=2048) |
|---|---|---|---|---|---|---|
| Cluster | 1x1x1 | 5328.56778 | 2719.77511 | 1382.56556 | 704.32679 | 360.64294 |
| ScaLAPACK | 1x1x2 | 5276.74287 | 2755.58509 | 1395.48266 | 718.42770 | 368.24621 |
| Band | 2x2x2 | 21031.96431 | 10686.37446 | 5586.12815 | 2781.05698 | 1431.91243 |

- Febcc-Col_k30x30x30.dat

  Calculation of conductivities and dielectric functions for bcc iron with $30 \times 30 \times 30$ k-points.

- Mn12.dat

  Calculation of conductivities and dielectric functions for a Mn12 molecular magnet with $1 \times 1 \times 1$ k-points.

- Si2_k10x10x10.dat

  Calculation of conductivities and dielectric functions for a silicon bulk with $10 \times 10 \times 10$ k-points.

- Si2_k1xk1xk1.dat

  Calculation of conductivities and dielectric functions for a silicon bulk with $1 \times 1 \times 1$ k-points.

Table 19: Total time of calculating conductivity and dielectric function of $VO_2$ (R phase, 384 atoms, $4 \times 4 \times 4$ supercell, k-grid=$1 \times 1 \times 1$). The speed-up ratio with repect to the case with 128 CPU cores is shown in the last column.

| # of CPUs | Total time of calculating conductivity and dielectric function (s) | Ratio of total time by # of CPUs to total time by 128 CPUs |
|---|---|---|
| 128 | 5328.56778 | 1.000 |
| 256 | 2719.77511 | 1.959 |
| 512 | 1382.56556 | 3.854 |
| 1024 | 704.32679 | 7.565 |
| 2048 | 360.64294 | 14.775 |

- VO2R-k16xk16x16.dat

  Calculation of conductivities and dielectric functions for VO2-R phase with $16 \times 16 \times 16$ k-points.

## 59.7 Automatic running test

To check whether the functionality of the conductivity and dielectric function calculations is properly installed or not, an automatic running test for the functionality can be performed by

### For the MPI parallel running

```
% mpirun -np 112 ./openmx -runtestCDDF
```

### For the MPI/OpenMP parallel running

```
% mpirun -np 56 ./openmx -runtestCDDF -nt 2
```

Then, OpenMX will run with five test cases, and compare calculated results with the reference results which are stored in 'work/cddf_example'. The comparison (absolute difference in the dielectric function) is stored in a file 'runtestCDDF.result' in the directory 'work'. The reference results were calculated using a Xeon cluster machine. If the difference is less than 0.1, we may consider that the installation is successful.

# 60 Interface with BoltzTraP

OpenMX is interfaced with BoltzTraP [99] which calculates electron transport coefficients based on the Boltzmann theory from the wave number dependence of the energy eigenvalues in the Kohn-Sham equation. The interface [100] with BoltzTraP enables us to calculate physical properties such as the Seebeck coefficient, electrical conductivity, electronic thermal conductivity, and the Hall coefficient. The functionality is compatible with not only the collinear calculations, but also the non-collinear calculations. When you publish a paper using the functionality, we would like to appreciate your citation of Ref. [100]. The interface with BoltzTraP2 will be released in the next version. The interface, which bridges between OpenMX and BoltzTraP, can be used by copying MX_TRAP.sh which is stored in the directory 'source' to the directory 'work' and executing it. As an example, let us introduce a calculation of non-doped Si bulk. One can perform the SCF calculation using an input file 'Si_BoltzTraP.dat' which is available in the directory 'work' as follows:

```
% mpirun -np 28 ./openmx Si_BoltzTraP
```

After the SCF calculation finishes normally, you obtain the out file 'Si_BoltzTraP.out'. Then, you need to copy MX_TRAP.sh which is stored in the directory 'source' to the directory 'work'. After that, you can perform MX_TRAP.sh as follows:

```
% sh MX_TRAP.sh
```

Then, the name of the 'out' files in the directory is listed as may be shown below.

```
=========================Outputfile List==========================
Si_BoltzTraP.out
==================================================================

Please enter the outputfile name ; Outputfile name =
```

Then, please enter the file name of the 'out' file after "Outputfile name =" and execute it by typing enter. Here we take Si_BoltzTraP.out as an example, and you may enter as

```
Please enter the output file name: Outputfile name = Si_BoltzTraP.out
```

When you press the enter key, the following message will be displayed:

```
..
....
Nospin::kloop kx ky kz:1681/1686
Nospin::kloop kx ky kz:1682/1686
Nospin::kloop kx ky kz:1683/1686
Nospin::kloop kx ky kz:1684/1686
Nospin::kloop kx ky kz:1685/1686
Nospin::kloop kx ky kz:1686/1686
.energy file for BoltzTraP has been generated.

.struct file for BoltzTraP has been generated.
```

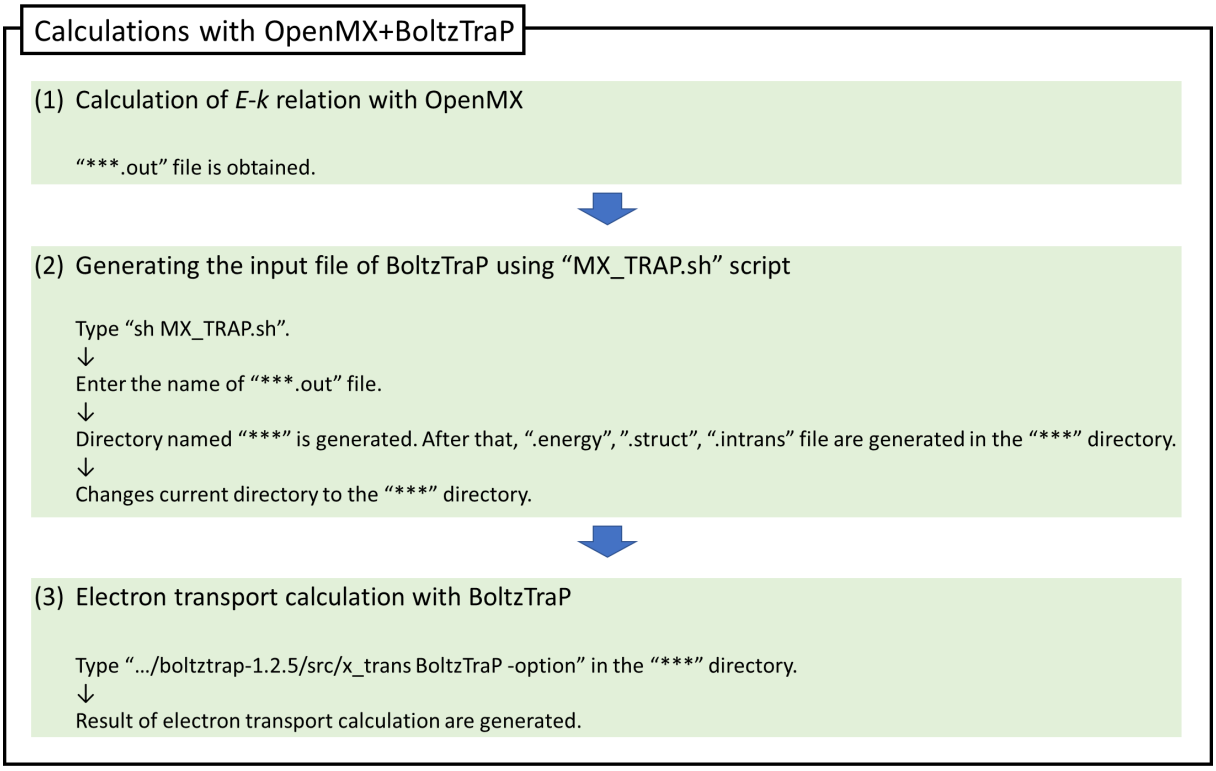



Figure 85: Computational protocol of the electron transport caculation by OpenMX and BoltzTraP.

```
.intrans file for BoltzTraP has been generated.

Conversion has been finished.

Directory is Si_BoltzTraP
```

After the above message is displayed, a directory named 'Si_BoltzTraP' is created, and four files such as 'Si_BoltzTraP.energy','Si_BoltzTraP.intrans', and 'Si_BoltzTraP.struct' will be stored in the directory as shown below:

```
% cd Si_BoltzTraP
% ls
Si_BoltzTraP.out Si_BoltzTraP.energy Si_BoltzTraP.struct Si_BoltzTraP.intrans
```

The '.out' file is a copy of the out file which was analyzed by the conversion. The '.energy' file is a file of energy eigenvalues. If the keyword 'scf.SpinPolarization' in the '.out' file is 'ON', then the '.energyup' and ".energydn" files are output, while '.energyso' file is output if the keyword 'scf. SpinPolarization' in the '.out' file is 'NC'. The '.struct' file is a file of unit lattice vectors. The unit is converted automatically into atomic units. The '.intrans' file describes input parameters necessary for electronic transport calculation. For details, please refer to the pdf manual enclosed with BoltzTraP package. The numerical value of the first item in the third line is the chemical potential $\mu$ calculated by OpenMX (unit is Ryd). The value of 10 in the fifth line is the Fourier interpolation factor of the band, which is set to 10 by default. The numerical value in the eighth row is the electron temperature of the

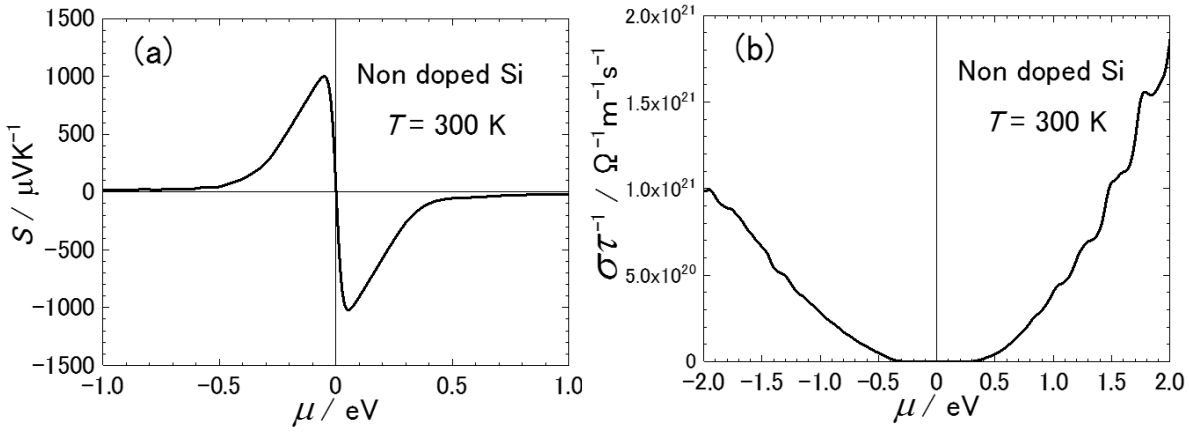



Figure 86: (a) Seebeck coefficient $S$, and (b) the electric conductivity $\sigma\tau_{\rm el}^{-1}$ of non-doped Si in the diamond structure as a function of the chemical potential at 300 K obtained by OpenMX and BoltzTraP. The Fourier interpolation factor was set to 25. The input file 'Si_BoltzTraP.dat' used for the calculation is available in the directory 'work'.

Fermi distribution. The first item is the maximum temperature. The second item is the temperature step size when calculating the temperature dependence. If the temperature dependence is not to be calculated, then enter the same value as the first item. By default, the temperature specified by 'scf.ElectronicTemperature' in the '.out' file is entered. If 'scf.ElectronicTemperature' is not specified, $T = 300K$ will be assigned.

Next, please move to the directory 'Si_BoltzTraP' and type BoltzTraP as shown below. The procedure from this point is the same as the normal BoltzTraP calculation. Therefore, please refer to the PDF manual enclosed in the BoltzTraP package for details.

```
% cd Si_BoltzTraP
% 'path to BoltzTrap'/boltztrap-1.2.5/sr/x_trans BoltzTraP
```

If the stored energy eigenvalue file is '.energyup', '.energydn' or '.energyso', please add '-up', '-dn', or '-so' as an option after x_trans BoltzTraP, and execute it. Then, electron transport calculations are performed for the energy eigenvalue file specified by the option. When the calculation is completed normally, the following message will be displayed:

```
========================== BoltzTraP vs 1.2.5 =============
99.786 u 0.076 s 1: 40.04 99.8\% 0 + 0 k 0 + 1612 0 io 0 pf + 0 w
```

The average values of the electron transport properties over the $x$-, $y$-, and $z$-axes are stored in 'Si_BoltzTraP.trace.', while the electron transport properties of the $x$-, $y$-, and $z$-axes are stored in 'Si_BoltzTraP.condtens'. Figure 85 summarizes the computational protocol of the electron transport calculation by OpenMX and BoltzTraP. In Fig. 86 we show the Seebeck coefficient $S$ and chemicafnl potential $\mu$ dependence of electrical conductivity $\sigma\tau_{\rm el}^{-1}$ of non-doped Si bulk. The input file 'Si_BoltzTraP.dat' used for the calculation is available in the directory 'work'.

Extensitve benchmark calculations using the functionality are found in the supplementary material: '11664_2017_6020_MOESM1_ESM.pdf' of Ref. [100]. Among the benchmark calculations we show in

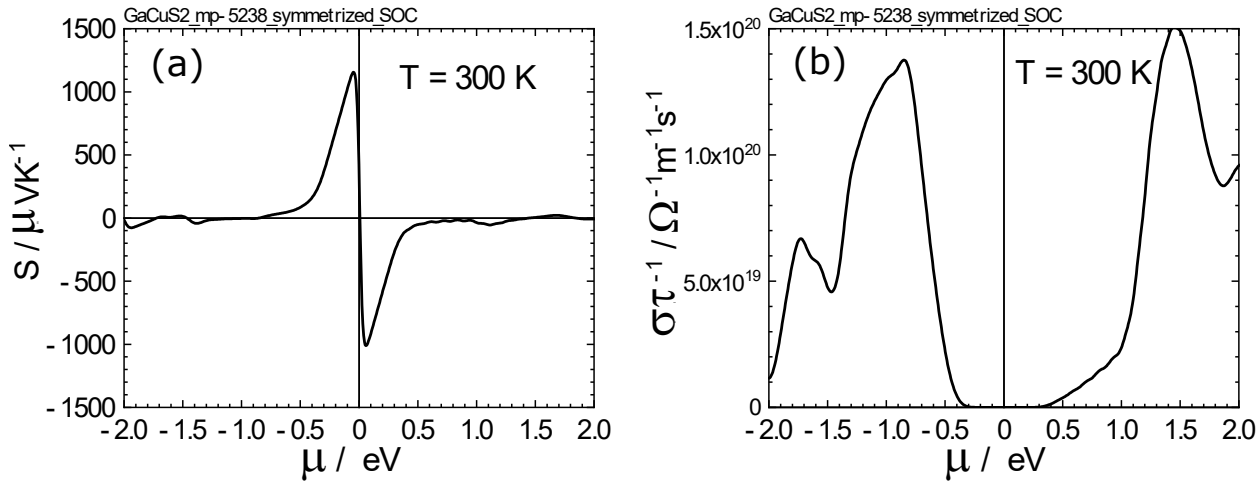



Figure 87: (a) Seebeck coefficient $S$, and (b) electric conductivity $\sigma\tau_{\rm el}^{-1}$ of $GaCuS_2$ bulk as a function of the chemical potential at 300 K obtained by OpenMX and BoltzTraP. The Fourier interpolation factor was set to 10. The input file 'GaCuS2_mp-5238_symmetrized_SOC.dat' used for the calculation is available in the directory 'work'.

Fig. 87 a computational result for $GaCuS_2$ bulk in which the non-collinear calculation with spin-orbit coupling was performed. The input file 'GaCuS2_mp-5238_symmetrized_SOC.dat' used for the calculation is available in the directory 'work'.

# 61 Calculation of Energy vs. lattice constant

## 61.1 Energy vs. lattice constant

The calculation of Energy vs. lattice constant is supported by the following keywords:

```
MD.Type                      EvsLC      #
MD.EvsLC.Step                 0.4       # default=0.4%
MD.maxIter                    32        # default=1
MD.EvsLC.flag               1 1 1       # default=1 1 1
# (0: fixed, 1:expansion, -1:contraction)
```

When 'MD.Type' is set to 'EvsLC', the total energy is calculated step by step by changing unit cell vectors, **a**, **b**, and **c**. The change of unit cell vectors is done uniformly by expanding them by a percentage, where the reference is the initial vectors, specified with 'MD.EvsLC.Step'. The number of steps is specified by the keyword 'MD.maxIter'. If you want to fix some of lattice vectors, the keyword 'MD.EvsLC.flag' is available, where the default setting is '1 1 1' corresponding to the uniform expansion of **a**-, **b**-, **c**-axes, respectively. The flag '0' means no change of the corresponding axis, and '-1' the uniform contraction. After the calculation, you will obtain a file '*System.Name*.EvsLC', where '*System.Name*' is 'System.Name'. The columns in the file '*System.Name*.EvsLC' are arranged in order of $a_x$, $a_y$, $a_z$, $b_x$, $b_y$, $b_z$, $c_x$, $c_y$, $c_z$ in Å, and the total energy in Hartree, where $a(b,c)_x$, $a(b,c)_y$, and $a(b,c)_z$ are $x$-, $y$-, and $z$-coordinates of the **a**(**b**,**c**) vector, respectively. As an example, calculation of Energy vs. lattice for the fcc Mn bulk is shown in Fig. 88, where the equilibrium lattice constant and bulk modulus were evaluated by fitting the data to the Murnaghan equation of state with a code 'murn.f' provided on the website [148].

## 61.2 Delta factor

As well as 'EvsLC', a similar functionality is provided as

```
MD.Type                      DF
```

by which OpenMX automatically calculates the total energy of the system with volumes of -6, -4, -2, 0, 2, 4, and 6 %, where the original structure given in the input file is taken to be the reference. The regulation of volume is simply performed by considering uniform change of lattice vectors, **a**-, **b**-, and **c**-axes. The volume and the corresponding total energy are output to a file '*System.Name*.DF'. The data can be used to calculate the delta factor proposed in Ref. [40].

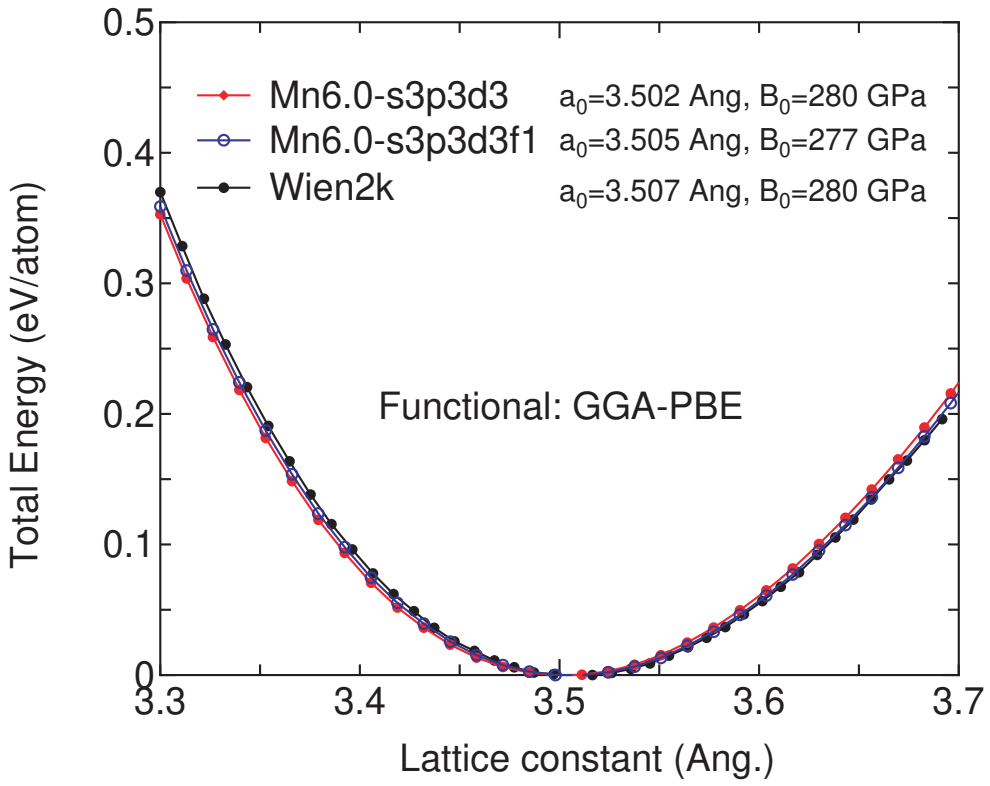



Figure 88: Total energy vs. lattice constant for the fcc Mn bulk calculated by the keyword 'EvsLC'. The input file used for the calculation is 'Mnfcc-EvsLC.dat' in the directory 'work'.

## 62 Fermi surface

The Fermi surface is visualized by XCrySDen [105] and FermiSurfer [143, 144]. When you perform calculations of the density of states by the following keywords:

```
Dos.fileout                on          # on|off, default=off
Dos.Erange              -20.0  20.0    # default = -20 20
Dos.Kgrid                 61 61 61     # default = Kgrid1 Kgrid2 Kgrid3
FermiSurfer.fileout       on           # default = off, on/off
```

you will obtain a file '*System.Name*.FermiSurf0.bxsf' which is readable by XCrySDen [105] and a file '*System.Name*.FermiSurf_s0_a*A*.frmsf' which is readable by FermiSurfer [143, 144], where '*System.Name*' is 'System.Name', and *A* is the serial number of atoms. To obtain the files readable by FermiSurfer, you need to switch on the keyword 'FermiSurfer.fileout'. As well as 'Dos.Fileout', 'DosGauss.fileout' can also be used for the purpose, while the files readable by FermiSurfer are not generated in the case of 'DosGauss.fileout'. In case of spin-polarized calculations, for XCrySDen two files '*System.Name*.FermiSurf0.bxsf' and '*System.Name*.FermiSurf1.bxsf' are generated for spin-up and spin-down states, respectively, and for FermiSurfer two files '*System.Name*.FermiSurf_s0_a*A*.frmsf' and '*System.Name*.FermiSurf_s1_a*A*.frmsf' are generated for spin-up and spin-down states, respectively. The data file for FermiSurfer can be used to display the color plot of the contribution from each atoms. For example, if we plot the character of atomic orbitals of the atom 1, we read '*Sys-*

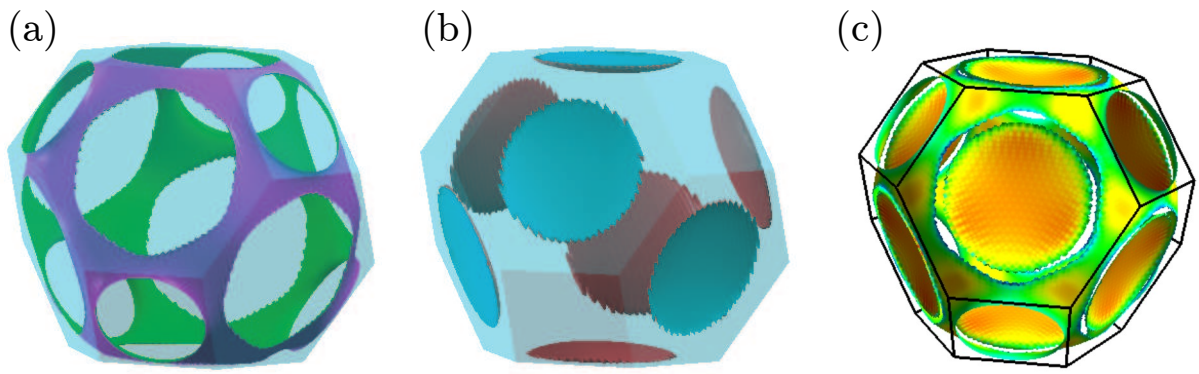



Figure 89: Fermi surfaces of the fcc Ca bulk visualized by XCrySDen [105] in (a) and (b). Since two sorts of bands intersect with the Fermi energy (chemical potential), two Fermi surfaces are shown in (a) and (b). (c) The color plot of the Fermi velocity of the material: the plot is visualized by using FermiSurfer [143, 144]. The input file used for the calculation is 'Cafcc_FS.dat' in the directory 'work'.

*tem.Name*.FermiSurf_s0_a1.frmsf'. In case of non-collinear calculations, a file '*System.Name*.FermiSurf.bxs' and '*System.Name*.FermiSurf_a*A*.frmsf' are generated. It is noted that a large number of **k**-points should be used in order to obtain a smooth Fermi surface. As an example, Fermi surfaces of the fcc Ca bulk are shown in Fig. 89. The input file used for the calculation is available as 'Cafcc_FS.dat' in the directory 'work'.

# 63 Analysis of difference in two Gaussian cube files

A utility tool is provided to generate a Gaussian cube file which stores the difference between two Gaussian cube files for total charge density, spin density, and potentials. If you analyze the difference between two states, this tool would be useful.

**(1) Compiling of diff_gcube.c**

There is a file 'diff_gcube.c' in the directory 'source'. Compile the file as follows:

```
% gcc diff_gcube.c -lm -o diff_gcube
```

When the compile is completed normally, then you can find an executable file 'diff_gcube' in the directory 'source'. Please copy the executable file to the directory 'work'.

**(2) Calculation of the difference**

If you want to know the difference between two Gaussian cube files 'input1.cube' and 'input2.cube', and output the result to a file 'output.cube', then perform the executable file as follows:

```
% ./diff_gcube input1.cube input2.cube output.cube
```

The difference is output to 'output.cube' in the Gaussian cube format. Thus, you can easily visualize the difference using many software, such XCrySDen [105], VESTA [103], and Molekel [104]. In fact, Fig. 28 in the Section 'Electric field' was made by this procedure.

# 64 Analysis of difference in two geometrical structures

A utility tool is provided to analyze the difference between two geometrical coordinates in two xyz files which store Cartesian coordinates. The following three analyses are supported: a root mean square of deviation (RMSD) between two Cartesian coordinates defined by

$$\mathrm{RMSD} = \sqrt{\frac{\sum_i^{N_{\mathrm{atom}}} (R_i - R_i^0)^2}{N_{\mathrm{atom}}}}$$

a mean deviation (MD) between two Cartesian coordinates defined by

$$\mathrm{MD} = \frac{\sum_i^{N_{\mathrm{atom}}} |R_i - R_i^0|}{N_{\mathrm{atom}}}$$

and a mean deviation between bond lengths (MDBL) defined by

$$\mathrm{MDBL} = \frac{\sum_i^{N_{\mathrm{bond}}} |BL_i - BL_i^0|}{N_{\mathrm{bond}}}$$

where $N_{\mathrm{atom}}$ and $N_{\mathrm{bond}}$ are the number of atoms and the number of bonds with bond length (BL) within a cutoff radius. Also, the deviation vector between xyz coordinate of each atom is output to a xsf file 'dgeo_vec.xsf' in the XCrySDen format. If you analyze the difference between two geometries, this tool would be useful.

### (1) Compiling of diff_gcube.c

There is a file 'diff_gcube.c' in the directory 'source'. Compile the file as follows:

```
% gcc diff_geo.c -lm -o diff_geo
```

When the compile is completed normally, then you can find an executable file 'diff_geo' in the directory 'source'. Please copy the executable file to the directory 'work'.

### (2) Calculation of the difference

You can find the following usage in the header part of diff_geo.c.

```
usage:
./diff_geo file1.xyz file2.xyz -d rmsd

option
-d rmsd      a root mean square of deviation
-d md        a mean deviation
-d mdbl 2.2  a mean deviation between bond lengths,
2.2 (Ang) means a cutoff bond length which
can be taken into account in the calculation
```

If you want to know RMSD between two Cartesian coordinates, run as follows:

```
% ./diff_geo file1.xyz file2.xyz -d rmsd
```

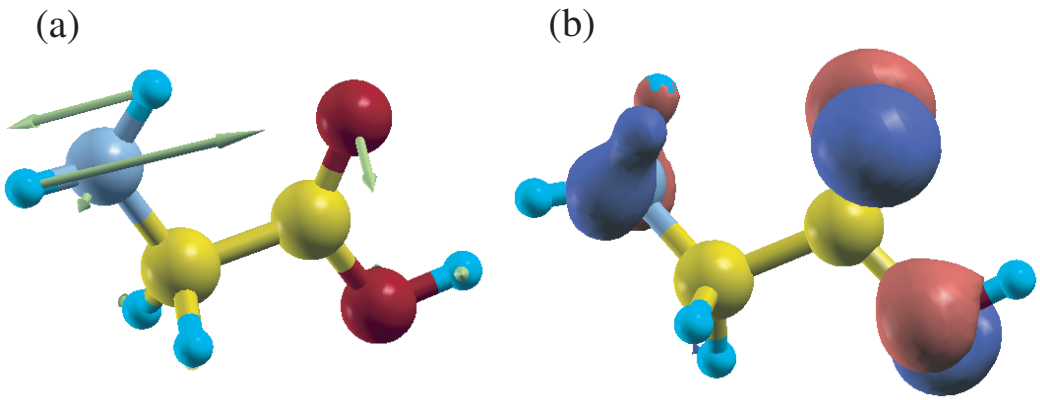



Figure 90: (a) Vectors corresponding to the deviation of atomic coordinates in optimized structures and (b) the difference of total charge density between a neutral and one electron doped glycine molecule. These figures were visualized by XCrySDen. In Fig. (b) blue and red colors indicate the decrease and increase of total charge density, respectively.

The calculated result appears in the standard output (your display). Also, a xsf file 'dgeo_vec.xsf' is generated in the XCrySDen format, which stores the difference between Cartesian coordinates of each atom in a vector form. This file can be visualized using 'Display→Forces' in XCrySDen. When MDBL is calculated, please give a cutoff bond length (Å). Bond lengths below the cutoff bond length are taken into account for the RMSD calculation. Figure 90 shows vectors corresponding to the deviation of atomic coordinates in optimized structures and the difference of total charge density between a neutral and one electron doped glycine molecule. We see that the large structural change seems to take place together with the large charge deviation. This example illustrates that the tool would be useful when we want to know how the structure is changed by the charge doping and the electric field.

# 65 Analysis of difference charge density induced by the interaction

The redistribution of charge (spin) density induced by the interaction between two systems A and B can be analyzed by the following procedure:

**(i) Calculate the composite system consisting of A and B**

Then, you will have a cube file for charge (spin) density. Let it be 'AB.cube'. Also, you will find 'Grid_Origin' in the standard output which gives $x$-, $y$-, and $z$-components of the origin of the regular grid as:

```
Grid_Origin  xxx  yyy  zzz
```

The values will be used in the following calculations (ii) and (iii).

**(ii) Calculate the system A**

This calculation must be performed by the same calculation condition with the same unit cell as in the composite system consisting of A and B. Also, the coordinates of the system A must be the same as in the calculation (i). To use the same origin as in the calculation (i) rather than the use of an automatically determined origin, you have to include the following keyword in your input file:

```
scf.fixed.grid  xxx  yyy  zzz
```

where 'xxx yyy zzz' is the coordinate of the origin you got in the calculation (i). Then, you will have a cube file for charge (spin) density. Let it be 'A.cube'.

**(iii) Calculate the system B**

As well as the calculation (ii), this calculation must be performed by the same calculation condition with the same unit cell as in the composite system consisting of A and B. Also, the coordinates of the system B must be the same as in the calculation (i). To use the same origin as in the calculation (i) rather than the use of an automatically determined origin, you have to include the following keyword in your input file:

```
scf.fixed.grid  xxx  yyy  zzz
```

where 'xxx yyy zzz' is the coordinate of the origin you got in the calculation (i). Then, you will have a cube file for charge (spin) density. Let it be 'B.cube'.

**(iv) Compile two codes**

compile two codes as follows:

```
% gcc diff_gcube.c -lm -o diff_gcube
% gcc add_gcube.c -lm -o add_gcube
```

**(v) Generate a cube file for difference charge (spin) density**

First, generate a cube file for the superposition of two charge (spin) densities of the systems A and B by

```
% ./add_gcube A.cube B.cube A_B.cube
```

The file 'A_B.cube' is the cube file for the superposition of charge (spin) density of two isolated systems. Then, you can generate a cube file for the difference charge (spin) density induced by the interaction as follows:

```
% ./diff_gcube AB.cube A_B.cube dAB.cube
```

The file 'dAB.cube' is the cube file for the difference charge (spin) density induced by the interaction, where the difference means (AB - A_B).

## 66 Automatic determination of the cell size

When you calculate an isolated system, you are required to provide a super cell so that the isolated system does not overlap with the image systems in the repeated cells via basis orbitals. The larger cell size can cause a numerical inefficiency, since a larger number of grids are used in the solution of the Poisson's equation in this case. Therefore, the use of the minimum cell size is desirable in terms of computational efficiency. OpenMX supports the requirement. If you remove the specification for the cell size, that is, from '<Atoms.UnitVectors' to 'Atoms.UnitVectors>', then OpenMX automatically determines an appropriate cell which does not overlap the next cells and fulfills the required cutoff energy. The determined cell vectors are displayed in the standard output like this:

```
<Set_Cluster_UnitCell> automatically determined UnitCell(Ang.)
<Set_Cluster_UnitCell> from atomic positions and Rc of PAOs (margin= 10.00%)
<Set_Cluster_UnitCell> 6.614718 0.000000 0.000000
<Set_Cluster_UnitCell> 0.000000 6.041246 0.000000
<Set_Cluster_UnitCell> 0.000000 0.000000 6.614718

widened unit cell to fit energy cutoff (Ang.)
A = 6.744142 0.000000 0.000000 (48)
B = 0.000000 6.322633 0.000000 (45)
C = 0.000000 0.000000 6.744142 (48)
```

# 67 Interface for developers

An interface for developers is provided. The matrix elements for the Hamiltonian, overlap, density matrix, position operator, and momentum operator, which are all obtained from the SCF calculation, can be accessed from your post processing code using the functionality. It should be also noted that from the OpenMX Ver. 3.9, the data format for the relevant codes 'SCF2File.c', 'read_scfout.c', 'read_scfout.h', and 'analysis_example.c' have been changed so that the full information of density matrix, and the matrices for the position operator and momentum operator can be included in the scfout file. So, the scfout files generated by the older versions of OpenMX cannot be analyzed by Ver. 3.9. These data can be utilized by the following steps:

1. Generation of 'HS.fileout'

   Include the keyword 'HS.fileout' in your input file as follows:

   ```
   HS.fileout                    on      # on|off, default=off
   ```

   and perform a convetional OpenMX calculation. Then, these data are output to a file '*System.Name*.scfout' where *System.Name* means System.Name in your input file.

2. make analysis_example

   In the directory 'source' compile by

   ```
   % make analysis_example
   ```

   Then, an executable file 'analysis_example' is generated in the directory 'work'.

3. ./analysis_example *System.Name*.scfout

   Move to the directory 'work', and then perform the program as follows:

   ```
   % ./analysis_example System.Name.scfout
   or
   % ./analysis_example System.Name.scfout > HS.out
   ```

   You can find the elements of the Hamiltonian, the overlap, and the density matrices in a file 'HS.out'

4. explanation of analysis_example

   In a file 'analysis_example.c' you can find a detailed description for these data. A part of the description is as follows:

   ```
   ***********************************************************************
   You can utilize a filename.scfout which is generated by the SCF
   calculation of OpenMX by the following procedure:

   1. Define your main routine as follows:
   ```

```
int main(int argc, char *argv[])

2. Include a header file, "read_scfout.h", in your main routine
(if you want, also in other routines) as follows:

#include "read_scfout.h"

3. Call a function, read_scfout(), in the main routine as follows:

read_scfout(argv);
*****************************************************************
```

It is noted that 'polB', 'jx', and 'kSpin' have been developed using the functionality as post processing code.

# 68 Calling OpenMX as library or computational engine

Note: I (Ozaki) had misunderstood how MPI_Comm_spawn works at the moment of the release of OpenMX Ver. 3.9 (Dec. 3, 2019). So, a part of the desctription below is not correct. Please use the capability at your own risk.

OpenMX can be utilized as library or computational engine from your program using MPI_Comm_spawn. You may mpirun your program and may want to call OpenMX with different input files from the program in different MPI groups at the same time. In such cases the functionality may be useful. Let us illustrate the functionality by introducing 'example_mpi_spawn.c' which is stored in the directory 'source'. The code 'example_mpi_spawn.c' can be compiled as

```
% make example_mpi_spawn
```

Then, you may obtain the executable file of 'example_mpi_spawn' in the directory 'work'. Please move to the directory 'work' and perform as

```
% mpirun -np 24 -machinefile machine0 --oversubscribe ./example_mpi_spawn
```

Setting the number of MPI processes, the machinefile, and the parameters may be changed depending on your computational environment. In some enviroment, you may need to attach '–oversubscribe' as option for mpirun in order to access the full physical cores on your machine. In the test calculation, the MPI processes of 24 are divided to three groups, each of which consists of 8 MPI processes, and in each group having 8 MPI processes OpenMX runs with an input file: 'Methane.dat', 'C60.dat', or 'Fe2.dat'. Starting from the example code as shown below, you may be able to develop a host program which calls OpenMX as child process. In the code 'Make_Comm_Worlds' creates three MPI communicaition groups, and 'MPI_Comm_spawn' calls OpenMX with one of the input files as argument. Note that OpenMX called by 'MPI_Comm_spawn' will not display most of standard output but write them in 'System.Name.std' to avoid appearence of too much information as standard output in such a case that OpenMX is multiply called by 'MPI_Comm_spawn' at the same time.

```
/************************************************************************

  example_mpi_spawn.c:

  An example of calling OpenMX as library by MPI_Comm_spawn, where
  a given MPI processes are grouped to three new MPI communication
  groups and OpenMX runs with an input file: 'Methane.dat', 'C60.dat',
  or 'Fe2.dat' in each MPI group.

  Log of example_mpi_spawn.c:

     25/Sep./2019  Released by Taisuke Ozaki

************************************************************************/

#include "mpi.h"
```

```
#include <stdio.h>
#include <stdlib.h>

void Make_Comm_Worlds(
   MPI_Comm MPI_Curret_Comm_WD,
   int myid0,
   int numprocs0,
   int Num_Comm_World,
   int *myworld1,
   MPI_Comm *MPI_CommWD,     /* size: Num_Comm_World */
   int *NPROCS1_ID,          /* size: numprocs0 */
   int *Comm_World1,         /* size: numprocs0 */
   int *NPROCS1_WD,          /* size: Num_Comm_World */
   int *Comm_World_StartID   /* size: Num_Comm_World */
   );

int main(int argc, char *argv[])
{
  int i,j;
  int numprocs0,myid0,ID0;
  int numprocs1,myid1;
  int num;
  int Num_Comm_World1;
  int myworld1;
  int *NPROCS1_ID,*NPROCS1_WD;
  int *Comm_World1;
  int *Comm_World_StartID;
  MPI_Comm *MPI_CommWD;
  MPI_Comm comm;
  MPI_Comm *intercomm;

  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&numprocs0);
  MPI_Comm_rank(MPI_COMM_WORLD,&myid0);

  /* set Num_Comm_World1 */

  Num_Comm_World1 = 3;

  /* allocation of arrays */

  NPROCS1_ID = (int*)malloc(sizeof(int)*numprocs0);
  Comm_World1 = (int*)malloc(sizeof(int)*numprocs0);
  NPROCS1_WD = (int*)malloc(sizeof(int)*Num_Comm_World1);
```

```
Comm_World_StartID = (int*)malloc(sizeof(int)*Num_Comm_World1);
MPI_CommWD = (MPI_Comm*)malloc(sizeof(MPI_Comm)*Num_Comm_World1);
intercomm = (MPI_Comm*)malloc(sizeof(MPI_Comm)*Num_Comm_World1);

/* Make_Comm_Worlds */

Make_Comm_Worlds(MPI_COMM_WORLD, myid0, numprocs0, Num_Comm_World1,
                 &myworld1, MPI_CommWD,
                 NPROCS1_ID, Comm_World1, NPROCS1_WD, Comm_World_StartID);

/* get numprocs1 and myid1 */

MPI_Comm_size(MPI_CommWD[myworld1],&numprocs1);
MPI_Comm_rank(MPI_CommWD[myworld1],&myid1);

/*
printf("numprocs0=%2d myid0=%2d myworld1=%2d numprocs1=%2d myid1=%2d\n",
        numprocs0,myid0,myworld1,numprocs1,myid1);
*/

/* MPI_Comm_spawn */

char command[] = "./openmx";
char **argvin;
char *inputfiles[] = { "Methane.dat", "C60.dat", "Fe2.dat" };

argvin=(char **)malloc(2 * sizeof(char *));
argvin[0] = inputfiles[myworld1];
argvin[1] = NULL;

MPI_Comm_spawn( command, argvin, numprocs1, MPI_INFO_NULL, 0,
                MPI_CommWD[myworld1], &intercomm[myworld1], MPI_ERRCODES_IGNORE );

/* MPI_Barrier */

MPI_Barrier(MPI_COMM_WORLD);

/* freeing of arrays */

free(NPROCS1_ID);
free(Comm_World1);
free(NPROCS1_WD);
free(Comm_World_StartID);
free(MPI_CommWD);
```

```
  free(intercomm);

  /* MPI_Finalize() */

  fflush(stdout);
  MPI_Finalize();

  return 0;
}


void Make_Comm_Worlds(
   MPI_Comm MPI_Curret_Comm_WD,
   int myid0,
   int numprocs0,
   int Num_Comm_World,
   int *myworld1,
   MPI_Comm *MPI_CommWD,     /* size: Num_Comm_World */
   int *NPROCS1_ID,          /* size: numprocs0 */
   int *Comm_World1,         /* size: numprocs0 */
   int *NPROCS1_WD,          /* size: Num_Comm_World */
   int *Comm_World_StartID   /* size: Num_Comm_World */
   )
{
  int i,j,is,ie;
  int ID0;
  int numprocs1,myid1;
  int num;
  double avnum;
  int *new_ranks;
  MPI_Group new_group,old_group;

.....

}
```

# 69 Automatic force tester

An effective way of assuring the reliability of implementation of many functionalities is to compare analytic and numerical forces. If any program bug is introduced, they will not be consistent with each other. To do this, one can run an automatic tester by

### For serial running

```
% ./openmx -forcetest 0
```

### For parallel running

```
% ./openmx -forcetest 0 "mpirun -np 4 openmx"
```

where '0' is a flag to specify energy terms to be included in the consistency check, and one can change 0 to 8. Each number corresponds to

```
flag           0   1   2   3   4   5   6   7   8

Kinetic        1   0   1   0   0   0   0   0   0
Non-local      1   0   0   1   0   0   0   0   0
Neutral atom   1   0   0   0   1   0   0   0   0
diff Hartree   1   0   0   0   0   1   0   0   0
Ex-Corr        1   0   0   0   0   0   1   0   0
E. Field       1   0   0   0   0   0   0   1   0
Hubbard U      1   0   0   0   0   0   0   0   1
```

where '1' means that it is included in the force consistency check. In a directory 'work/force_example', there are 36 test inputs which are used for the force consistency check. After finishing the test, a file 'forcetest.result' is generated in the directory 'work'. You will see results of the comparison as follows:

```
force_example/C2_GGA.dat
flag= 0
Numerical force= -(Utot(s+ds)-Utot(s-ds))/(2*ds)
ds=   0.0003000000
Forces (Hartree/Bohr) on atom 1
x              y               z
Analytic force       -1.676203071292 -1.397113794193 -1.117456296887
Numerical force      -1.676101156844 -1.397036485449 -1.117288361652
diff                 -0.000101914447 -0.000077308744 -0.000167935235

force_example/C2_LDA.dat
flag= 0
Numerical force= -(Utot(s+ds)-Utot(s-ds))/(2*ds)
......
....
```

# 70 Automatic memory leak tester

In OpenMX, the memory used is dynamically allocated when it is required. However, the dynamic memory allocation causes often a serious memory leak which wastes the memory used as the MD steps increase. To check the memory leak, one can run OpenMX as follows:

### For serial running

```
% ./openmx -mltest
```

### For parallel running

```
% ./openmx -mltest "mpirun -np 4 openmx"
```

By monitoring VSZ and RSS actually used at the same monitoring point in the program code for 14 test inputs in a directory 'work/ml_example', one can find whether the memory leak takes place or not. After finishing the run, a file 'mltest.result' is generated in the directory 'work'. You will see the monitored VSZ and RSS as a function of MD steps as follows:

```
   1     ml_example/DIA8.dat

                  CPU (%)     VSZ (kbyte)    RSS (kbyte)
MD_iter=   1      68.300     397396         123076
MD_iter=   2      96.400     436264         131916
MD_iter=   3      99.000     436264         131916
MD_iter=   4      97.900     436264         131916
MD_iter=   5      98.800     436264         131916
MD_iter=   6      99.300     436264         131916
MD_iter=   7      98.800     436264         131916
MD_iter=   8      99.200     436264         131916
MD_iter=   9      99.500     436264         131916
MD_iter=  10      99.100     436264         131916
MD_iter=  11      99.400     436264         131916
MD_iter=  12      99.500     436264         131916
MD_iter=  13      99.300     436260         131916
MD_iter=  14      99.500     436264         131916
MD_iter=  15      99.300     436264         131916
MD_iter=  16      99.500     436260         131916
MD_iter=  17      99.600     436264         131916
MD_iter=  18      99.400     436264         131916
MD_iter=  19      99.600     436264         133848
MD_iter=  20      99.400     436264         133848
MD_iter=  21      99.500     436264         133848
MD_iter=  22      99.600     436264         133848
MD_iter=  23      99.500     436264         133848
MD_iter=  24      99.500     436264         133848
```

```
MD_iter=  25      99.700     436264         133848
MD_iter=  26      99.500     436264         133848
MD_iter=  27      99.600     436264         133848
MD_iter=  28      99.500     436264         133848
MD_iter=  29      99.600     436264         133848
MD_iter=  30      99.600     436264         133848


   2     ml_example/DIA8_DC.dat

                  CPU (%)     VSZ (kbyte)    RSS (kbyte)
MD_iter=   1     101.000     412508         136448
MD_iter=   2     100.000     516940         210312
MD_iter=   3      98.200     517016         210440
MD_iter=   4      98.900     517016         210440
MD_iter=   5      99.300     517016         210440
MD_iter=   6      99.500     517016         210440
   ......
   ....
```

# 71 Analysis of memory usage

The memory usage can be found by analyzing files '*.memory0', '*.memory1',..., and '*.memory#', where '*' is the file name specified by the keyword 'System.Name' and the number in the file extension corresponds to process ID in the MPI parallelization. The files are output by setting the keyword 'memory.usage.fileout' as

```
memory.usage.fileout   on   # default=off, on|off
```

As an example 'met.memory0' is shown below

```
Memory: SetPara_DFT: Spe_PAO_XV                                0.01 MBytes
Memory: SetPara_DFT: Spe_PAO_RV                                0.01 MBytes
Memory: SetPara_DFT: Spe_Atomic_Den                            0.01 MBytes
Memory: SetPara_DFT: Spe_PAO_RWF                               0.57 MBytes
Memory: SetPara_DFT: Spe_RF_Bessel                             1.03 MBytes
Memory: SetPara_DFT: Spe_VPS_XV                                0.01 MBytes
Memory: SetPara_DFT: Spe_VPS_RV                                0.01 MBytes
Memory: SetPara_DFT: Spe_Vna                                   0.01 MBytes
Memory: SetPara_DFT: Spe_VH_Atom                               0.01 MBytes
Memory: SetPara_DFT: Spe_Atomic_PCC                            0.01 MBytes
Memory: SetPara_DFT: Spe_VNL                                   0.11 MBytes
Memory: SetPara_DFT: Spe_VNLE                                  0.00 MBytes
Memory: SetPara_DFT: Spe_VPS_List                              0.00 MBytes
.....
....
...
Memory: Poisson: array0                                        4.00 MBytes
Memory: Poisson: array1                                        4.00 MBytes
Memory: Poisson: request_send                                  0.00 MBytes
Memory: Poisson: stat_send                                     0.00 MBytes
Memory: Poisson: request_recv                                  0.00 MBytes
Memory: Poisson: stat_recv                                     0.00 MBytes
Memory: Force: Hx                                              0.00 MBytes
Memory: Force: Hy                                              0.00 MBytes
Memory: Force: Hz                                              0.00 MBytes
Memory: Force: CDM0                                            0.00 MBytes
Memory: Data_Grid_Copy_B2C_1: Work_Array_Snd_Grid_B2C          0.72 MBytes
Memory: Data_Grid_Copy_B2C_1: Work_Array_Rcv_Grid_B2C          0.72 MBytes
Memory: total                                                256.99 MBytes
```

The file can be obtained by setting the keyword in the input file 'Methane.dat' and performing a single process. Note that memory usages for most of arrays are listed in the file, but the list is not complete.

## 72 Output of large-sized files in binary mode

Large-scale calculations produce large-sized files in text mode such as cube files. The IO access to output such files can be very time consuming in machines of which IO access is not fast. In such a case, it is better to output those large-sized files in binary mode. The procedure is supported by the following keyword:

```
OutData.bin.flag    on    # default=off, on|off
```

Then, all large-sized files will be output in binary mode. The default is 'off'.

The output binary files are converted using a small code 'bin2txt.c' stored in the directory 'source' which can be compiled as

```
gcc bin2txt.c -lm -o bin2txt
```

As a post processing, you will be able to convert as

```
./bin2txt *.bin
```

The functionality will be useful for machines of which IO access is not fast.

# 73 Converting of Gaussian cube format to periodic XSF format

When we use a volumetric data-visualization software (such as VESTA and XCrysDen), the acceptable operation is dependent on the data format and the keyword. The current version of VESTA (at 2015/8) cannot display a periodic structure with the Gaussian cube format input file; we have to use a periodic XSF format-input file to display a periodic structure. OpenMX produces Gaussian cube format files of the charge/spin density, Kohn-Sham orbitals, eiggenchannels, and so on; A tool (`cube2xsf`) to convert the Gaussian cube format to the periodic XSF format is prepared.

The usage of it is as follows:

```
$ cube2xsf a.cube
$ cube2xsf b.cube.bin
$ cube2xsf a.cube b.cube.bin
$ cube2xsf *.cube.bin
```

Then, files that have `.xsf` extention are generated.

# 74 Examples of the input files

For your convenience, the input files of examples shown in the manual are available in the directory 'work' as listed below:

```
Molecules or clusters
C60.dat             SCF calc. of a C60 molecule
C60_DC.dat          DC calc. of a C60 molecule
CG15c_DC.dat        DC calc. of DNA
Cr2_CNC.dat         Constrained DFT calc. of a Cr2 dimer
Doped_NT.dat        SCF calc. of doped carbon nanotube
Fe2.dat             SCF calc. of a Fe2 dimer
Gly_NH.dat          Nose-Hoover MD of a glycine molecule
Gly_VS.dat          Velocity scaling MD of a glycine molecule
H2O.dat             Geometry opt. of a water molecule
MCCN.dat            DC calc. of a a multiply connected carbon nanotube
Methane2.dat        Geometry opt. of a distorted methane molecule
Methane.dat         SCF calc. of a  methane molecule
Methane_OO.dat      Orbital optimization of a methane molecule
Methane_ED.dat      Total energy decomposition of a methane molecule
Mn12.dat            SCF calc. of a single molecular magnet, Mn12
Mol_MnO_NC.dat      Non-collinear SCF calc. of a MnO molecule
Nitro_Benzene.dat   SCF calc. of a nitro benzene molecule under E-field
Pt13.dat            SCF calc. of a Pt13 cluster
Pt63.dat            SCF calc. of a Pt63 cluster
SialicAcid.dat      SCF calc. of a sialic acid molecule
Valorphin_DC.dat    DC calc. of valorphin molecule
Valorphin_MO.dat    Molecular obital calculations
C2H4_NEB.dat        NEB calc. of C2H4 dimer
C60_LO.dat          Low-order scaling calc. of a C60 molecule
Fe_Cluster_jx.dat   jx calculation of a Fe dimer
H2O+0.dat           SCF calc. of a neutral water molecule
H2O+1.dat           SCF calc. of a plus 1 water molecule
C2H2.dat            Initial state calculation of C2H2 for XPS
C2H2-CH.dat         Final state calculation of C2H2 for XPS

Bulk
Cdia.dat            SCF calc. of bulk diamond
MnO_NC.dat          Non-collinear SCF calc. of bulk MnO
FeO_NC.dat          Non-collinear SCF calc. of bulk FeO
CoO_NC.dat          Non-collinear SCF calc. of bulk CoO
NiO_NC.dat          Non-collinear SCF calc. of bulk NiO
Crys-NiO.dat        SCF calc. of bulk NiO
NiO-cFLL.dat        LDA+U calc. of NiO by the cFLL scheme
NiO-sFLL.dat        LDA+U calc. of NiO by the sdcFLL scheme
NiO-Yukawa.dat      LDA+U calc. of NiO to estimate J and F^4/F^2
DIA64_Band.dat      SCF calc. of bulk diamond including 64 atoms
DIA8_DC.dat         DC calc. of bulk diamond including 8 atoms
DIA64_DC.dat        DC calc. of bulk diamond including 64 atoms
DIA216_DC.dat       DC calc. of bulk diamond including 216 atoms
DIA512_DC.dat       DC calc. of bulk diamond including 512 atoms
DIA512-1.dat        Krylov O(N) calc. of bulk diamond including 512 atoms
Febcc2.dat          SCF calc. of bcc Fe
Fefcc-SpinSpiral.dat  SCF calc. of fcc Fe for spin spiral calculation
GaAs.dat            Non-collinear calc. of bulk gallium arsenide
```

```
NaCl.dat           SCF calc. of bulk NaCl
NaCl_FC.dat        SCF calc. of bulk NaCl with a Cl-site vacancy
Si8.dat            Geometry opt. of distorted Si bulk
Si8-pV.dat         Enthalpy opt. of Si bulk under 10 GPa
Si8-LNO.dat        O(N) DC-LNO calc. of silicon crystal including 8 atom
NdCo5_4f.dat       E vs. lattice constant calc. of NcCo4 bulk including the 4f states
NdCo5_4f+U.dat     E vs. lattice constant calc. of NcCo4 bulk including the 4f states with plus U
NdCo5_OC.dat       E vs. lattice constant calc. of NcCo4 bulk with a Nd open core pseudopotential
Al-Si111_ESM.dat   ESM calc. of Al-Si interface
Cafcc_FS.dat       Fermi surface calc. of the fcc Ca bulk
Graphite_STM.dat   STM image of graphene
Mnfcc-EvsLC.dat    E vs. lattice constant calc. of the fcc Mn bulk
Si8_NEB.dat        NEB calc. for hydrogen in Si
DIA8-VA.dat        Virtual atom SCF calc. of diamond crystal
FePt.dat           SCF calc. of L10-FePt within collinear DFT
FePt-NC-SCF.dat    SCF calc. of L10-FePt within non-collinear DFT
FePt-NC.dat        One-shot diagonalization of L10-FePt with SOI
Fe_Bulk_jx.dat     jx calculation of bcc Fe
GaCuS2_mp-5238_symmetrized_SOC.dat  SCF calculation for BoltzTrap calculation
Si_BoltzTraP.dat   SCF calculation of Si bulk for BoltzTrap calculation
Si2_k50x50x50.dat  Optical conductivity for Si bulk
TiC216.dat         Initial state calculation of TiC for XPS
TiC216-CH3.dat     Final state calculation of TiC for XPS
Si-4-SOI.dat       Initial state calculation of Si for XPS
Si-4-CH-SOI1.dat   Final state (j=3/2) calculation of Si for XPS
Si-4-CH-SOI6.dat   Final state (j=1/2) calculation of Si for XPS
Au111Surface_FL.dat  spin texture analysis of Au111 by the FermiLoop scheme
Au111Surface_GC.dat  spin texture analysis of Au111 by the GridCalc scheme
Au111Surface_BD.dat  spin texture analysis of Au111 by the BandDispersion scheme
Au111Surface_MO.dat  spin texture analysis of Au111 by the MulPOnly scheme
SiC_Primitive_BD.dat spin texture analysis of SiC by the BandDispersion scheme
```

# 75 Known problems

- Overcompleteness of basis functions

  When a large number of basis functions is used for dense bulk systems with fcc, hcp, and bcc like structures, the basis set tends to be overcomplete. In such a case, you may observe erratic eigenvalues. To avoid the overcompleteness, a small number of optimized basis functions should be used. Another way to avoid the problem is to switch off the keyword 'scf.ProExpn.VNA' as

  ```
  scf.ProExpn.VNA      off       # on|off, default = on
  ```

  In this case, you may need to increase the cutoff energy for the numerical grid in real space by the keyword 'scf.energycutoff'.

- Difficulty in getting the SCF convergence

  For large-scale systems with a complex (non-collinear) magnetic structure, a metallic electric structure, or the mixture, it is quite difficult to get the SCF convergence. In such a case, one has to mix the charge density very slowly, indicating that the number of SCF steps to get the convergence becomes large unfortunately.

- Difficulty in getting the optimized structure

  For weak interacting systems such as molecular systems, it is not easy to obtain a completely optimized structure, leading that the large number of iteration steps is required. Although the default value of criterion for geometrical optimization is $10^{-4}$ Hartree/Bohr for the largest force, it would be a compromise to increase the criterion from $10^{-4}$ to $5 \times 10^{-4}$ in such a case.

# 76 OpenMX Forum

For discussion of technical issues on OpenMX and ADPACK, there is a forum (http://www.openmx-square.org/forum/patio.cgi). It is expected that the forum is utilized for sharing tips in use of OpenMX and for further code development. Points of concern for use of this forum can be found in http://www.openmx-square.org/forum/note.html

# 77 Other sources of information about OpenMX

Several websites provide information related to OpenMX as listed below. We hope that you may find a proper information depending on your tastes.

- Lecture materials at Oregon State University.
  http://physics.oregonstate.edu/ tatej/COURSES/ph575/doku.php?id=openmx
- Lecture materials at Tokyo Institute of Technology
  https://www.slideshare.net/cms_initiative/open-mx-lecture
- Lecture materials at Kanazawa University
  http://f-ishii.w3.kanazawa-u.ac.jp/ja/index.cgi?page=%B7%D7%BB%BB%CA%AA%C0%AD%B2%CA%B3%D82018
- Tutorial by HPCI
  http://www.hpci-office.jp/pages/appli_openmx
- Tutorial by Dr. Toyoda
  https://sites.google.com/site/mtoyodacmp/openmx-memo
- Tutorial by Dr. Ito
  http://www-fps.nifs.ac.jp/ito/memo/openmx04.html
- Information by MateriApps
  https://ma.issp.u-tokyo.ac.jp/app/594
- Information by Dr. Inukai
  https://www5.hp-ez.com/hp/calculations/page114
- Tutorial material by Prof. Kato
  https://www.slideshare.net/cms_initiative/materiapps-openmx
- Benchmark by Dr. Larsson
  https://www.nsc.liu.se/ pla/blog/2014/06/11/openmx/
- Benchmark by HPC Technologies
  https://www.hpc-technologies.co.jp/openmx-benchmarks

# 78 Linkage to other tools

Linkage to other tools is summarized here.

- Trial use

  Some of you might want the quick trial use of OpenMX. The following is one of such tools.

  - MateriApps LIVE!:
    http://ma.cms-initiative.jp/en/whats-materiapps/try_apps/about-materiapps-live?set_language=en
    MateriApps LIVE! offers an environment where one can try out computational materials science simulation freely, using a notebook PC, etc. All environment required to begin tutorials, such as MateriApps applications, OS (Debian GNU/Linux), editors, and visualization tools, is provided in a USB memory stick. Since OpenMX is available as one of simulation tools in MateriApps LIVE!, you might be able to consider MateriApps LIVE! as an environment for the trial use of OpenMX.

- Binary distribution

  The binary distribution of OpenMX on LINUX environments is available as follows.

  - Debian: https://packages.debian.org/search?keywords=openmx
  - Ubuntu: https://launchpad.net/ubuntu/+source/openmx

- Graphical User Interface (GUI) and/or job scheduling environment

  A couple of GUIs and job scheduling environments for OpenMX calculations are available as follows.

  - ASE: https://wiki.fysik.dtu.dk/ase/ase/calculators/openmx.html
  - sisl: http://zerothi.github.io/sisl/docs/latest/api-generated/sisl.io.html
  - OMXTool: https://github.com/Ncmexp2717/OMXTool
  - Winmostar: https://winmostar.com/jp/manual_jp/V9/html/winmos/solid/winmos_openmx.html

- Visualization in general

  OpenMX generates cube, md, xyz, xsf, axsf, and cif files. These files can be visualized by many software. The following is some of them.

  - OpenMX Viewer: http://www.openmx-square.org/viewer/
  - XCrySDen: http://www.xcrysden.org/
  - VESTA: http://jp-minerals.org/vesta/en/
  - Molekel: http://www.cscs.ch/molekel/

- Visualization of Fermi surfaces

  The Fermi surfaces can be visualized by FermiSurfer.

  - FermiSurfer: http://fermisurfer.osdn.jp/

- Analysis of molecular dynamics simulations

  You might want to analyze the trajectory generated by molecular dynamics simulations in OpenMX. The following is one of such tools.

  - ASAP: http://www.mch.rwth-aachen.de/

- A tool to read and operate OpenMX Kohn-Sham Hamiltonian

  A tool by Dr. Artem Pulkin is available to read and operate OpenMX Kohn-Sham Hamiltonian at the following website:

  - openmx-hks: https://github.com/pulkin/openmx-hks

- Tight Binding Studio

  Software package to construct Tight Binding (TB) model in combination with first-principles calculations including the OpenMX code.

  - Tight Binding Studio: https://tight-binding.com/

- Thermoelectric properties

  Thermoelectric properties can be calculated by BoltzTraP via an interface with OpenMX or an external tool: QTWARE based on an NEGF method.

  - BoltzTraP: https://doi.org/10.1016/j.cpc.2006.03.007
  - QTWARE: http://www.rs.tus.ac.jp/takahiro/QTWare.html

- Jx: An open source software for calculating magnetic interactions based on magnetic force theory, which is interfaced with OpenMX code. Though the original OpenMX code also supports a similar calculation, the Jx code is a post processing code which has been developed by the Prof. M.J. Han group in KAIST, and released independently.

  - Jx: https://doi.org/10.1016/j.cpc.2019.106927

- Physical properties derived from Wannier functions

  Maximally localized Wannier functions can be utilized to efficiently and accurately calculate a wide variety of physical properties. OpenMX provides an interface with an post processing code: Wannier90.

  - Wannier90: http://wannier.org/

    The post processing code enables us to constructs maximally localized Wannier functions using results from a first-principles SCF calculation, and calculates physical properties such as Wannier projected DOS and bandstructure, Fermi surface, Berry phase related properties (anomalous Hall conductivity, optical conductivity, and orbital magnetization), and thermoelectric properties. See the details for the section 'Interface with Wannier90'.

- Phonon related properties

  To calculate phonon related properties such as phonon dispersion and thermal conductivity, OpenMX can be combined with ALAMODE as explained in the following website:

  - ALAMODE: https://alamode.readthedocs.io/en/latest/tutorial.html

- DCore: DMFT solver interfaced with DFT codes

  DCore based on dynamical mean-field theory (DMFT) has an interfaces with OpenMX as explained in the following website:

  - DCore: https://issp-center-dev.github.io/DCore/master/tutorial/srvo3_openmx/openmx.html#

# 79 Others

## Program

The program package is written in the C and F90 languages, including one makefile

`makefile,`

34 header files

```
BandDispersion.h Circular_Search.h EigenValue_Problem.h Eigen_HH.h GetOrbital.h
Inputtools.h Tools_BandCalc.h Tools_Search.h exx.h exx_debug.h exx_def_openmx.h
exx_file_eri.h exx_file_overlap.h exx_index.h exx_interface_openmx.h
exx_log.h exx_rhox.h exx_step1.h exx_step2.h exx_vector.h exx_xc.h f77func.h
jx.h jx_LNO.h jx_config.h jx_quicksort.h jx_tools.h jx_total_mem.h
lapack_prototypes.h mimic_sse.h openmx_common.h read_scfout.h tran_prototypes.h
tran_variables.h
```

and 350 routines

```
ADIIS_Mixing_DM.c ADenBand.c Allocate_Arrays.c AngularF.c BandDispersion.c Band_DFT_Col.c
Band_DFT_Col_NEGF.c Band_DFT_Col_Optical_ScaLAPACK.c Band_DFT_Dosout.c Band_DFT_MO.c
Band_DFT_NonCol.c Band_DFT_NonCol_GB.c Band_DFT_NonCol_Optical.c Band_DFT_kpath.c
Band_DFT_kpath_LNO.c Band_Dispersion.c Bench_MatMul.c BentNT.c BroadCast_ComplexMatrix.c
BroadCast_ReMatrix.c Calc_optical.c Circular_Search.c Cluster_DFT_Col.c Cluster_DFT_Dosout.c
Cluster_DFT_LNO.c Cluster_DFT_NonCol.c Cluster_DFT_ON2.c Cluster_DFT_OptOrb.c
Cluster_DFT_Optical.c Cluster_DFT_Optical_ScaLAPACK.c Cluster_DFT_ScaLAPACK.c Cont_Matrix0.c
Cont_Matrix1.c Cont_Matrix2.c Cont_Matrix3.c Cont_Matrix4.c Contract_Hamiltonian.c
Contract_iHNL.c Coulomb_Interaction.c Cutoff.c DFT.c DFTD3vdW_init.c DFTDvdW_init.c
DIIS_Mixing_DM.c DIIS_Mixing_Rhok.c Divide_Conquer.c Divide_Conquer_Dosout.c
Divide_Conquer_LNO.c DosMain.c Dr_KumoF.c Dr_RadialF.c Dr_VH_AtomF.c Dr_VNAF.c
EGAC_DFT.c Eff_Hub_Pot.c EigenBand_lapack.c EigenValue_Problem.c Eigen_HH.c Eigen_PHH.c
Eigen_PReHH.c Eigen_lapack.c Eigen_lapack2.c Eigen_lapack3.c Embedded_GFM.c EulerAngle_Spin.c
FT_NLP.c FT_PAO.c FT_ProExpn_VNA.c FT_ProductPAO.c FT_VNA.c FermiLoop.c File_CntCoes.c
Find_CGrids.c Force.c Force_test.c Free_Arrays.c Fuzzy_Weight.c GR_Pulay_DM.c Gaunt.c
Gauss_Legendre.c Generate_Wannier.c Generating_MP_Special_Kpt.c GetOrbital.c
Get_Cnt_Orbitals.c Get_Cnt_dOrbitals.c Get_OneD_HS_Col.c Get_Orbitals.c Get_dOrbitals.c
GridCalc.c Hamiltonian_Band.c Hamiltonian_Band_NC.c Hamiltonian_Band_NC_Hs2.c
Hamiltonian_Cluster.c Hamiltonian_Cluster_Hs.c Hamiltonian_Cluster_NC.c
Hamiltonian_Cluster_NC_Hs2.c Hamiltonian_Cluster_SO.c Hamiltonian_NC_Hs2.c
Init_List_YOUSO.c Initial_CntCoes.c Initial_CntCoes2.c Input_std.c Inputtools.c
Inputtools_kSpin.c Kerker_Mixing_Rhok.c Krylov.c KumoF.c LNO.c LU_inverse.c
Lapack_LU_inverse.c MD_pac.c MTRAN_EigenChannel.c Make_Comm_Worlds.c Make_FracCoord.c
Make_InputFile_with_FinalCoord.c Maketest.c Matrix_Band_LNO.c Memory_Leak_test.c
Merge_LogFile.c Mio_tester.c Mio_tester2.c Mixing_DM.c Mixing_H.c Mixing_V.c
MulPCalc.c MulPOnly.c Mulliken_Charge.c NBO_Cluster.c NBO_Krylov.c NabraMatrixElements.c
Nonlocal_Basis.c Nonlocal_RadialF.c Occupation_Number_LDA_U.c Opt_Contraction.c
OpticalConductivityMain.c Orbital_Moment.c OutData.c OutData_Binary.c Output_CompTime.c
Output_Energy_Decomposition.c Overlap_Band.c Overlap_Band_NC_Ss2.c Overlap_Cluster.c
Overlap_Cluster_LNO.c Overlap_Cluster_NC_Ss2.c Overlap_Cluster_Ss.c PhiF.c
Poisson.c Poisson_ESM.c Population_Analysis_Wannier.c Population_Analysis_Wannier2.c
Pot_NeutralAtom.c PrintMemory.c PrintMemory_Fix.c QuickSort.c RF_BesselF.c
RadialF.c ReLU_inverse.c RestartFileDFT.c Runtest.c SCF2File.c SetPara_DFT.c
Set_Aden_Grid.c Set_Allocate_Atom2CPU.c Set_ContMat_Cluster_LNO.c Set_CoreHoleMatrix.c
Set_Density_Grid.c Set_Hamiltonian.c Set_Initial_DM.c Set_Nonlocal.c Set_OLP_Kin.c
```

```
Set_OLP_p.c Set_Orbitals_Grid.c Set_ProExpn_VNA.c Set_Vpot.c Set_XC_Grid.c
Set_dOrbitals_Grid.c Show_DFT_DATA.c SigmaEK.c Simple_Mixing_DM.c Smoothing_Func.c
Spherical_Bessel.c Stress.c Stress_test.c TRAN_Add_ADensity_Lead.c TRAN_Add_Density_Lead.c
TRAN_Allocate.c TRAN_Allocate_NC.c TRAN_Apply_Bias2e.c TRAN_Band.c TRAN_Band_Col.c
TRAN_CDen_Main.c TRAN_Calc_CentGreen.c TRAN_Calc_CentGreenLesser.c TRAN_Calc_CurrentDensity.c
TRAN_Calc_GridBound.c TRAN_Calc_Hopping_G.c TRAN_Calc_OneTransmission.c TRAN_Calc_SelfEnergy.c
TRAN_Calc_SurfGreen.c TRAN_Calc_SurfGreen_Sanvito.c TRAN_Channel_Functions.c TRAN_Channel_Output.c
TRAN_Check_Input.c TRAN_Check_Region.c TRAN_Check_Region_Lead.c TRAN_Credit.c TRAN_DFT.c
TRAN_DFT_Dosout.c TRAN_DFT_NC.c TRAN_Deallocate_Electrode_Grid.c TRAN_Deallocate_RestartFile.c
TRAN_Distribute_Node.c TRAN_Input_std.c TRAN_Input_std_Atoms.c TRAN_Input_std_Atoms0.c
TRAN_Input_std_Atoms2.c TRAN_Main_Analysis.c TRAN_Main_Analysis_NC.c TRAN_Output_HKS.c
TRAN_Output_HKS_Write_Grid.c TRAN_Output_Trans_HS.c TRAN_Poisson.c TRAN_Print.c TRAN_Print_Grid.c
TRAN_Read.c TRAN_RestartFile.c TRAN_Set_CentOverlap.c TRAN_Set_CentOverlap_NC.c
TRAN_Set_Electrode_Grid.c TRAN_Set_IntegPath.c TRAN_Set_MP.c TRAN_Set_SurfOverlap.c
TRAN_Set_SurfOverlap_NC.c TRAN_Set_Value.c TRAN_adjust_Ngrid.c Tetrahedron_Blochl.c Timetool.c
Tools_BandCalc.c Tools_Search.c Total_Energy.c Unfolding_Bands.c VH_AtomF.c VNAF.c
Voronoi_Charge.c Voronoi_Orbital_Moment.c XANES0.c XC_CA_LSDA.c XC_Ceperly_Alder.c XC_EX.c
XC_PBE.c XC_PW92C.c Z2FH.c add_gcube.c analysis_example.c bandgnu13.c bin2txt.c calB.c
check_lead.c cube2xsf.c dampingF.c deri_dampingF.c diff_gcube.c diff_geo.c dtime.c esp.c
example_mpi_spawn.c expao.c exx.c exx_debug.c exx_file_eri.c exx_file_overlap.c exx_index.c
exx_interface_openmx.c exx_log.c exx_rhox.c exx_step1.c exx_step2.c exx_vector.c exx_xc.c
find_Emin.c find_Emin0.c find_Emin2.c find_Emin_withS.c frac2xyz.c gcube2oned.c gen_defile.c
init.c init_alloc_first.c intensity_map.c io_tester.c iterout.c iterout_md.c jx.c jx_LNO.c
jx_band_indiv.c jx_band_psum.c jx_cluster.c jx_config.c jx_quicksort.c jx_tools.c kSpin.c
lapack_dstedc1.c lapack_dstedc2.c lapack_dstedc3.c lapack_dstegr1.c lapack_dstegr2.c
lapack_dstegr3.c lapack_dsteqr1.c lapack_dstevx1.c lapack_dstevx2.c lapack_dstevx3.c
lapack_dstevx4.c lapack_dstevx5.c malloc_multidimarray.c md2axsf.c mimic_sse.c mpao.c
mpi_multi_world.c mpi_multi_world2.c mpi_non_blocking.c neb.c neb_check.c neb_run.c
openmx.c openmx_common.c outputfile1.c pdb2pao.c polB.c read_scfout.c readfile.c rmmpi.c
rot.c test_mpi.c test_mpi2.c test_mpi3.c test_mpi4.c test_openmp.c test_openmp2.c
test_openmp3.c tp.c truncation.c unit2xyz.c xyz2spherical.c zero_cfrac.c zero_fermi.c
elpa1.f90 get_elpa_row_col_comms.f90 solve_evp_complex.f90 solve_evp_real.f90
```

In addition, the following library packages are linked:

```
lapack,
blas,
fftw,
MPICH2 or OpenMP
```

### Copyright of the program package

The distribution of this program package follows the practice of the GNU General Public License version 3 (GPLv3) [102]. Moreover, the author, Taisuke Ozaki, possesses the copyright of the original version of this program package. We cannot offer any guarantee in your use of this program package. However, when you report program bugs, we will cooperate and work well as much as possible together with you to remove the problems.

### Acknowledgment

One of us (T.O.) would like to thank many colleagues in JRCAT, RICS-AIST, JAIST, and ISSP for helpful their suggestions and comments. One of us (T.O.) was partly supported by the following

national projects: SYNAF-NEDO [154], ACT-JST [155], NAREGI [156], CREST-JST [157], MEXT [158], CMSI [159], NEDO-ChouChou [160] and CDMSI [161]

## References

[1] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964); W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

[2] D. M. Ceperley and B. J. Alder, Phys. Rev. Lett., 45, 566(1980); J. P. Perdew and A. Zunger, Phys. Rev. B **23,** 5048 (1981).

[3] J. P. Perdew and A. Zunger, Phys. Rev. B **23,** 5048 (1981).

[4] J. P. Perdew and Y. Wang, Phys.Rev.B **45**, 13244 (1992).

[5] J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).

[6] A.E. Reed, L.A. Curtiss, and F. Weinhold, Chem. Rev. **88**, 899 (1988); E. D. Glendening, C. R. Landis, and F. Weinhold, WIREs Comput. Mol. Sci. **2**, 1 (2012).

[7] T. Ohwaki, M. Otani, and T. Ozaki, J. Chem. Phys. **140**, 244105 (2014).

[8] U. Von. Barth and L. Hedin, J. Phys. C: Solid State Phys. **5**, 1629 (1972).

[9] J. Kübler, K-H. Höck, J. Sticht, and A. R. Williams, J. Phys. F: Met. Phys. **18**, 469 (1988).

[10] J. Sticht, K-H. Höck, and J. Kübler, J. Phys.: Condens. Matter **1**, 8155 (1989).

[11] T. Oda, A. Pasquarello, and R.Car, Phys. Rev. Lett. **80**, 3622 (1998).

[12] A. H. MacDonald and S. H. Vosko, J. Phys. C: Solid State Phys. 12, 2977 (1979).

[13] Ph. Kurz, F. Forster, L. Nordstrom, G, Bihlmayer, and S. Blugel, Phys. Rev. B **69**, 024415 (2004).

[14] J. Harris, Phys. Rev. B **31**, 1770 (1985).

[15] R. D. King-Smith and D. Vanderbilt, Phys. Rev. B **47**, 1651 (1993).

[16] G. Theurich and N. A. Hill, Phys. Rev. B **64**, 073106 (2001).

[17] A. I. Liechtenstein, M. I. Katsnelson, V. P. Antropov, and V. A. Gubanov, J. Mag. Mag. Mat. **67**, 65 (1987).

[18] M. J. Han, T. Ozaki, and J. Yu, Phys. Rev. B **70**, 184421 (2004).

[19] A. Terasawa, M. Matsumoto, T. Ozaki, and Y. Gohda, J. Phys. Soc. Jpn. **88**, 114706 (2019).

[20] M. J. Han, T. Ozaki, and J. Yu, Phys. Rev. B **74**, 045110 (2006).

[21] S. Ryee and M.J. Han, J. Phys:Condens. Matter **30**, 275802 (2018).

[22] S. Ryee and M.J. Han, Scientific Reports **8**, 9559 (2018).

[23] http://www.openmx-square.org/tech_notes/tech6-1_0.pdf

[24] http://www.openmx-square.org/tech_notes/DFTU_notes_OpenMX.pdf

[25] S.L. Dudarev, G.A. Botton, S.Y. Savrasov, C.J. Humphreys, and A.P. Sutton, Phys. Rev. B **57**, 1505 (1998).

[26] A.I. Liechtenstein, V.I. Anisimov, and J. Zaanen, Phys. Rev. B **52**, R5467 (1995).

[27] A. N. Yaresko, V. N. Antonov, and P. Fulde, Phys. Rev. B **67**, 155103 (2003).

[28] L. Vaugier, H. Jiang, and S. Biermann, Phys. Rev. B **86**, 165105 (2012).

[29] F. Bultmark, F. Cricchio, O. Grånäs, and L. Nordström, Phys. Rev. B 80, 035121 (2009).

[30] L. V. Woodcock, Chem. Phys. Lett. **10** ,257 (1971).

[31] S. Nose, J. Chem. Phys. **81**, 511 (1984); S. Nose, Mol. Phys. **52**, 255 (1984); G. H. Hoover, Phys. Rev. A **31**, 1695 (1985)).

[32] G. B. Bachelet, D. R. Hamann, and M. Schluter, Phys. Rev. B **26**, 4199 (1982).

[33] N. Troullier and J. L. Martine, Phys. Rev. B **43**, 1993 (1991).

[34] L. Kleinman and D. M. Bylander, Phys. Rev. Lett. **48,** 1425 (1982).

[35] P. E. Blochl, Phys. Rev. B **41**, 5414 (1990).

[36] I. Morrison, D.M. Bylander, L. Kleinman, Phys. Rev. B **47**, 6728 (1993).

[37] D. Vanderbilt, Phys. Rev. B **41**, 7892 (1990).

[38] H.J. Monkhorst and J.D. Pack, Phys. Rev. B **13**, 5188 (1976).

[39] T. Auckenthaler, V. Blum, H.-J. Bungartz, T. Huckle, R. Johanni, L. Kraemer, B. Lang, and H. Lederer, P. R. Willems, Parallel Computing 27, 783 (2011).

[40] K. Lejaeghere, V. Van Speybroeck, G. Van Oost, and S. Cottenier, Critical Reviews in Solid State and Materials Sciences 39, 1 (2014); K. Lejaeghere et al., Science 351, aad3000 (2016).

[41] T. Ozaki, Phys. Rev. B. **67**, 155108, (2003); T. Ozaki and H. Kino, Phys. Rev. B **69**, 195113 (2004).

[42] T. Ozaki and H. Kino, Phys. Rev. B **72**, 045121 (2005).

[43] T. Ozaki, Phys. Rev. B **74**, 245101 (2006).

[44] T.V.T. Duy and T. Ozaki, Comput. Phys. Commun. 185, 777 (2014).

[45] T.V.T. Duy and T. Ozaki, Comput. Phys. Commun. 185, 153 (2014).

[46] S.F. Boys and F. Bernardi, Mol. Phys. **19**, 553 (1970).

[47] S. Simon, M. Duran, and J.J. Dannenberg, J. Chem. Phys. **105**, 11024 (1996).

[48] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias and J. Joannopoulos, Rev. Mod. Phys. **64**, 1045 (1992) and references therein.

[49] O. F. Sankey and D. J. Niklewski, Phys. Rev. B. **40**, 3979 (1989)

[50] W. Yang, Phys.Rev.Lett. **66**, 1438 (1991)

[51] T. Ozaki, M. Fukuda, G. Jiang, Phys. Rev. B **98**, 245137 (2018).

[52] J. Behler and M. Parrinello, Phys. Rev. Lett. **98**, 146401 (2007).

[53] V. Recoules and J.-P. Crocombette, Phys. Rev. B **72**, 104202 (2005).

[54] J. A. Anta and P. A. Madden, J. Phys.: Condens. Matter **11**, 6099 (1999).

[55] M. Kim, K. H. Khoo, and J. R. Chelikowsky, Phys. Rev. B **86**, 054104 (2012).

[56] P. Ordejon, E. Artacho, and J. M. Soler, Phys. Rev. B. **53**, 10441 (1996)

[57] D. R. Bowler and M. J. Gillan, Chem. Phys. Lett. **325**, 475 (2000).

[58] G. Kresse and J. Furthmeuller, Phys. Rev. B. **54**, 11169 (1996)

[59] G. P. Kerker, Phys. Rev. B **23**, 3082 (1981).

[60] T. A. Arias, M. C. Payne, and J. D. Joannopoulos, Phys. Rev. B **45**, 1538 (1992).

[61] D. Alfe, Comp. Phys. Commun. **118**, 32 (1999).

[62] P. Csaszar and P. Pulay, J. Mol. Struct. (Theochem) **114**, 31 (1984).

[63] J. Baker, J. Comput. Chem. **7**, 385 (1986)

[64] A. Banerjee, N. Adams, J. Simons, R. Shepard, J. Phys. Chem. **89**, 52 (1985)

[65] C. G. Broyden, J. Inst. Math. Appl. 6, 76 (1970); R. Fletcher, Comput. J. 13, 317 (1970); D. Goldrarb, Math. Comp. 24, 23 (1970); D. F. Shanno, Math. Comp. 24, 647 (1970).

[66] H.B. Schlegel, Theoret. Chim. Acta (Berl.) 66, 333 (1984); J.M. Wittbrodt and H.B. Schlegel, J. Mol. Struc. (Theochem) 398-399, 55 (1997).

[67] P. E. Blochl, O. Jepsen and O. K. Andersen, Phys. Rev. B **49**, 16223 (1994).

[68] The details of the implementation will be published elsewhere.

[69] A. D. Becke and R. M. Dickson, J. Chem. Phys. **89**, 2993 (1988).

[70] A. Svane and O. Gunnarsson, Phys. Rev. Lett. **65**, 1148 (1990).

[71] J. Tersoff and D. R. Hamann, Phys. Rev. B **31**, 805 (1985).

[72] G. Henkelman and H. Jonsson, J. Chem. Phys. **113**, 9978 (2000).

[73] T. Ozaki, K. Nishio, and H. Kino, Phys. Rev. **81**, 035116 (2010).

[74] T. Ozaki, Phys. Rev. B **75**, 035123 (2007).

[75] M. Brandbyge, J.-L. Mozos, P. Ordejon, J. Taylor, and K. Stokbro, Phys. Rev. B **65**, 165401 (2002)

[76] G. C. Liang, A. W. Ghosh, M. Paulsson, and S. Datta, Phys. Rev. B. **69**, 115302 (2004).

[77] H. Weng, T. Ozaki, and K. Terakura, Phys. Rev. B **79**, 235118 (2009).

[78] H. Kotaka, F. Ishii, and M. Saito, Jpn. J. Appl. Phys. 52, 035204 (2013).; N. Yamaguchi and F. Ishii, Appl. Phys. Express 10, 123003 (2017).

[79] T.B. Prayitno and F. Ishii, J. Phys. Soc. Jpn. **87**, 114709 (2018).

[80] T.B. Prayitno and F. Ishii, J. Phys. Soc. Jpn. **88**, 054701 (2019).

[81] T. Fukui, Y. Hatsugai and H. Suzuki J. Phys. Soc. Jpn. J. Phys. Soc. Jpn. **74**, 1674 (2005).

[82] W. Feng, J. Wen, J. Zhou, D. Xiao, and Y. Yao, Comput. Phys. Commun. **183**, 1849 (2012).

[83] T. Kato, H. Kotaka, and F. Ishii, JPS Conf. Proc. **5**, 011022 (2015).

[84] H. Sawahata, N. Yamaguchi, H. Kotaka, and F. Ishii, Jpn. J. Appl. Phys. 57, 030309 (2018).

[85] https://t-ozaki.issp.u-tokyo.ac.jp/meeting16/OMX-Sawahata-2016Nov.pdf

[86] L.M. Sandratskii, Adv. Phys. 47, 91 (1998).

[87] V.M. García-Suárez, C.M. Newman, C.J. Lambert, J.M. Pruneda, and J. Ferrer, J. Phys.: Condens. Matter **16**, 5453 (2004).

[88] T. Ozaki and C.C. Lee, Phys. Rev. Lett. **118**, 026401 (2017).

[89] J.F. Janak, Phys. Rev. B **18**, 7165 (1978).

[90] W.L. Jolly, K.D. Bomben, C.J. Eyermann, At. Data Nul. Data Tables **31**, 433 (1984).

[91] M. R. Jarvis, I. D. White, R. W. Godby, and M. C. Payne, Phys. Rev. B **56**, 14972 (1997).

[92] C. D. Wagner, W. M. Riggs, L. E. Davis, J. F. Moulder, and G. E. Mullenberg, Handbook of X-Ray Photoelectron Spectroscopy (Perkin-Elmer, Minnesota, 1979).

[93] C.-C. Lee, J. Yoshinobu, K. Mukai, S. Yoshimoto, H. Ueda, R. Friedlein, A. Fleurence, Y. Yamada-Takamura, and T. Ozaki, Phys. Rev. B **95**, 115437 (2017).

[94] C.-C Lee, B. Feng, M. D'angelo, R. Yukawa, R-Y Liu, T. Kondo, H. Kumigashira, I. Matsuda, and T. Ozaki, Phys. Rev. B **97**, 075430 (2018).

[95] K. Yamazaki, Y. Maehara, C-C Lee, J. Yoshinobu, T. Ozaki, and K. Gohara, J. Phys. Chem. C **122**, 27292 (2018).

[96] K.B.Snow and T.F. Thomas, Int. J. Mass Spectrom. Ion Processes **96**, 49 (1990).

[97] M. Ernzerhof and G.E. Scuseria, J. Chem. Phys. **110**, 5029 (1999).

[98] http://www.openmx-square.org/tech_notes/Dielectric_Function_YTL.pdf

[99] G.K.H. Madsen and D.J. Singh, Comput. Phys. Commun. 175, **67** (2006).

[100] M. Miyata, T. Ozaki, T. Takeuchi, S. Nishino, M. Inukai, and M. Koyano, J. Elec. Mater.**47**, 3254 (2017).

[101] https://www.imc.tuwien.ac.at//forschungsbereich_theoretische_chemie/forschungsgruppen /prof_dr_gkh_madsen_theoretical_materials_chemistry/boltztrap/

[102] http://www.gnu.org/

[103] http://jp-minerals.org/vesta/en/

[104] http://www.cscs.ch/molekel/

[105] http://www.xcrysden.org/

[106] T. Lis, Acta Crystallogra. B **36**, 2042 (1980).

[107] T. P. Davis T. J. Gillespie, F. Porreca, Peptides **10**, 747 (1989).

[108] A. Goldstein, S. Tachibana, L. I. Lowney, M. Hunkapiller, and L. Hood, Proc. Natl. Acad. Sci. U. S. A. **76**, 6666 (1979).

[109] U. C. Singh and P. A. Kollman, J. Comp. Chem. **5**, 129(1984).

[110] L. E. Chirlian and M. M. Francl, J. Com. Chem. **8**, 894(1987).

[111] B. H. Besler, K. M. Merz Jr. and P. A. Kollman, J. Comp. Chem. **11**, 431(1990).

[112] http://www.webelements.com/

[113] M. Cardona, N. E. Christensen, and G. Gasol, Phys. Rev. B **38**, 1806 (1988).

[114] G. Theurich and N. A. Hill, Phys. Rev. B **64**, 073106 (2001).

[115] *Physics of Group IV Elements and III-V Compounds*, edited by O.Madelung, M.Schulz, and H. Weiss, Landolt-Büornstein, New Series, Group 3, Vol. 17, Pt.a (Springer, Berlin, 1982).

[116] T. Ono and K. Hirose, Phys. Rev. B **72**, 085105 (2005).

[117] W. N. Mei, L. L. Boyer, M. J. Mehl, M. M. Ossowski, and H. T. Stokes, Phys. Rev. B **61**, 11425 (2000).

[118] I. V. Solovyev. A. I. Liechtenstein, K. Terakura, Phys. Rev. Lett. **80**, 5758.

[119] K. Knopfle, L. M. Sandratskii, and J. Kubler, J. Phys:Condens. Matter **9**, 7095 (1997).

[120] I. S. Dhillon and B. N. Parlett, SIAM J. Matrix Anal. Appl. **25**, 858 (2004).

[121] J. J. M. Cuppen, Numer. Math. **36**, 177 (1981); M. Gu and S. C. Eisenstat, SIAM J. Mat. Anal. Appl. **16**, 172 (1995).

[122] N. Mazari and D. Vanderbilt, Phys. Rev. B **56**, 12 847 (1997).

[123] I. Souza, N. Marzari and D. Vanderbilt, Phys. Rev. B **65**, 035109 (2001).

[124] T. Ozaki, Phys. Rev. B **82**, 075131 (2010).

[125] M. Otani and O. Sugino, Phys. Rev. B **73**, 115407 (2006).

[126] O. Sugino, I. Hamada, M. Otani, Y. Morikawa, T. Ikeshoji, and Y. Okamoto, Surf. Sci. **601**, 5237 (2007).

[127] M. Otani, I. Hamada, O. Sugino, Y. Morioka, Y. Okamoto, and T. Ikeshoji, J. Phys. Soc. Jpn. **77**, 024802 (2008).

[128] T. Ohwaki, M. Otani, T. Ikeshoji, and T. Ozaki, J. Chem. Phys. **136**, 134101 (2012).

[129] R.M. Eastment and C.H.B. Mee, J. Phys. F: Metal Phys. **3**, 1738 (1973).

[130] P.O. Gartland, S. Berge, and B.J. Slagsvold, Phys. Rev. Lett. **28**, 738 (1972).

[131] M. Chelvayohan and C.H.B. Mee, J. Phys. C: Solid State Phys. **15**, 2305 (1982).

[132] G.V. Hansson and S.A. Flodström, Phys. Rev. B **18**, 1572 (1978).

[133] G.D. Kubiak, J. Vac. Sci. Technol. A **5**, 731 (1987).

[134] G. Henkelman and H. Jonsson, J. Chem. Phys. **113**, 9978 (2000).

[135] S. Grimme, J. Comput. Chem. **27**, 1787 (2006).

[136] S. Grimme, J. Antony, S. Ehrlich, and H. Krieg, J. Chem. Phys. 132, 154104 (2010).

[137] S. Grimme, S. Ehrlich and L. Goerigk, J. Comput. Chem. 32, 1456 (2011).

[138] J.G. Hill, J.A. Platts, and H.-J. Werner, Phys. Chem. Chem. Phys. 8, 4072 (2006).

[139] C. Li, L. Wan, Y. Wei, and J. Wang, Nanotechnology **19**, 155401 (2008).

[140] L. Zhang, B. Wang, and J. Wang, Phys. Rev. B **84**, 115412 (2011).

[141] M. Paulsson and M. Brandbyge, Phys. Rev. B **76**, 115117 (2007).

[142] C.-C. Lee, Y. Yamada-Takamura, and T. Ozaki, J. Phys.: Condens. Matter **25**, 345501 (2013).

[143] M. Kawamura, Comp. Phys. Comm. **239**, 197 (2019).

[144] http://fermisurfer.osdn.jp/

[145] http://www.wannier.org/

[146] https://github.com/Ncmexp2717/OMXTool

[147] H. Makino, I. H. Inoue, M. J. Rozenberg, I. Hase, Y. Aiura, and S. Onari, Phys. Rev. B 58, 4384 (1998).

[148] http://www.fhi-berlin.mpg.de/th/fhi98md/Murn/readme_murn.html

[149] http://www.openmx-square.org/

[150] http://www.netlib.org/lapack/

[151] http://www.openmx-square.org/viewer/

[152] Y.-T Lee and T. Ozaki, Journal of Molecular Graphics and Modelling **89**, 192 (2019).

[153] http://www.nanotec.es/

[154] http://www.nanoworld.jp/synaf/

[155] http://act.jst.go.jp/

[156] http://ccinfo.ims.ac.jp/nanogrid/

[157] http://www.jst.go.jp/

[158] http://computics-material.jp/index-e.html

[159] http://www.cms-initiative.jp/ja

[160] A project commissioned by the New Energy and Industrial Technology Development Organization of Japan (NEDO) Grant (P16010).

[161] https://cdmsi.issp.u-tokyo.ac.jp/